International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 9 Issue 10, October – 2025, Pages: 7-18

Development of a Hybrid Method to Enhance Context Memory for a Chatbot Application Based on Large Language Models

Nadiia Bohdan, Iryna Tvoroshenko, Volodymyr Gorokhovatskyi, Oleg Kobylin
Department of Informatics
Kharkiv National University of Radio Electronics,
Kharkiv, Ukraine
e-mail: nadiia.bohdan@nure.ua; iryna.tvoroshenko@nure.ua; volodymyr.horokhovatskyi@nure.ua; oleg.kobylin@nure.ua

Abstract: *This paper addresses the critical challenge of maintaining long-term contextual coherence in LLM-based chatbots, particularly in scenarios demanding multi-stage reasoning and complex information recall. The core scientific novelty of this research lies in the development and justification of a Hybrid Methodology for Contextual Memory Enhancement. This methodology synergistically integrates the temporal continuity of recent messages' context with the precision recall capabilities of RAG and AI analysis agent, effectively mitigating the common issue of contextual dilution over extended dialogues. The research involved the development of chatbot application implementing this method and subsequent rigorous scenario-based testing, which validated the superior performance of the proposed hybrid approach. The results provide definitive recommendations for optimizing LLM memory management, paving the way for more robust and reliable conversational AI systems capable of advanced, multi-turn reasoning and complex task resolution.*

Keywords—artificial intelligence; chatbot; embeddings; generation; large language models; retrieval-augmented memory

1. INTRODUCTION

Contextual memory is a key component determining the coherence, sequence, and overall relevance of responses in Large Language Model (LLM)-based dialogue systems [1].

Despite the rapid growth in computational power, most contemporary LLMs face a fundamental limitation known as the limited context window: the size of the working memory the model can consider when generating a response.

While modern models can handle up to hundreds of thousands of tokens per session (e.g., beyond the original 4096 tokens of GPT-3.5) [2], this constraint often leads to contextual amnesia: the AI (artificial intelligence) cannot fully comprehend the human chat context, losing critical details mentioned in older messages. This results in the loss of important details, repetitive questions, contradictory answers, and a general decline in communication quality, becoming critical in scenarios demanding long-term role-playing or multi-stage technical consultations.

The researchers and developers propose a range of architectural solutions aimed at effectively managing and expanding contextual memory. Sliding Window Context method maintains recent messages in the context window.

While it is fast, computationally inexpensive, and ensures coherence in short dialogues, it suffers from the irreversible loss of information from the beginning of the chat, rendering it ineffective for long-term memory tasks. Retrieval-Augmented Generation (RAG) approach stores the entire dialogue history in a vector database and retrieves only the most semantically relevant fragments to augment the LLM's prompt.

RAG provides the highest accuracy and access to the full history, but its implementation is complex, requires significant computational resources, and often yields responses that are technically correct but lack the natural flow and immediate coherence of recent messages [3].

Furthermore, none of these methods, in isolation, solve two critical high-level challenges: the lack of specialized prompts for deep context analysis and the absence of automatic state updates (e.g., tracking the dynamics of relationships in role-playing scenarios), both of which are crucial for generating logically sound and natural responses.

This discrepancy forms the main research question: is it possible to create a methodology that combines the speed and natural coherence of Sliding Window Context with the depth and long-term accuracy of RAG, while also integrating an agent mechanism for sophisticated state management?

Thus, this work substantiates that creating a hybrid tandem of these opposing approaches can compensate for each other's drawbacks, resulting in a highly effective methodology.

The main disadvantage of Sliding Window Context (the loss of information) is completely nullified because RAG ensures the entire dialogue history is permanently stored in a vector database.

Conversely, the Sliding Window Context provides the LLM with fresh messages, ensuring natural and subtle contextual coherence in the immediate response [4-7]. The process of developing this method requires additional enhancements through the integration of AI agent analysis to influence the quality of the language model's understanding of the context, update dynamic states in real-time, and utilize effective prompts with metadata [4].

The object of research is the methods for improving the contextual memory of chatbots based on large language models.

The goal of the research is the development and empirical substantiation of a hybrid methodology for improving chatbot contextual memory, which integrates Sliding Window Context, RAG, and agent analysis to ensure long-term coherence and logical connectivity [8].

2. Related Works

To gain a comprehensive understanding of existing approaches to improving contextual memory, an in-depth analysis of scientific publications was conducted [9]. This section contains an extended review of literature sources related to the testing of the results of applying existing methods for improving the contextual memory of chatbots based on large language models, with an emphasis on their advantages, disadvantages, limitations, and practical value.

In the paper [6], the authors note that Sliding Window Context method is effective for short-term, focused conversations where information quickly loses its relevance. However, it's most significant and obvious drawback: the irreversible loss of information from the dialogue's outset: renders it critically limited in long-term, complex dialogue scenarios.

This finding is further confirmed in the comprehensive study [10], where Sliding Window Context demonstrated poor performance when the dialogue length exceeded the context window size [11-13].

In the research [14], the authors conducted comprehensive research and introduced a new dataset for evaluating the effectiveness of various memory management methods in long-term dialogues, focusing on methods such as Sliding Window Context and RAG. The researchers focused on providing an objective assessment of the performance of each approach in real-world conditions that require context retention over long exchanges. The authors state that the Sliding Window Context method shows worse results when the dialogue exceeds the context window size. Furthermore, the study showed that the RAG method significantly outperforms the "sliding window." However, the main drawback of the article [14] is its focus exclusively on studying memory management methods based on existing models and datasets. While the authors provide analysis, their work does not include the development of an optimal, implementable methodology that integrates the research findings. They do not propose a ready-to-implement algorithm that would allow developers to integrate such architecture into their own projects.

Conversely, the RAG method, originally explored in seminal work such as [5], has gained prominence as a solution for factual accuracy and long-term memory. RAG overcomes the primary limitation of traditional LLMs by coupling the generative model with an external knowledge base (a vector database), allowing access to the full history.

Reviewed experiments, including those in [10], confirm that RAG-models significantly surpass standard LLMs in the veracity and justification of responses, as it reduces the risk of hallucinations.

Crucially, RAG, while factually precise, often fails to maintain the natural, immediate conversational flow because it relies solely on semantically similar fragments [3].

The information retrieved by RAG is merely a set of old messages; the generative model does not receive specific instructions on how to logically integrate this information, often resulting in inconsistent and mechanical answers.

The papers [12] (on RAFT) and [13] explore enhancements, but the need for a unified architecture that strategically dictates how and when to apply memory fragments based on the dialogue's dynamic state remains.

In the article [15], the authors conduct an in-depth analysis of the ethical and practical challenges associated with implementing long-term memory in LLM-based dialogue systems.

Unlike previous sources that focused on efficiency and architecture, this work concentrates on the potential risks that arise when chatbots store conversation history. The primary goal of the study was to identify and classify these risks, as well as propose possible ways to minimize them [16, 17].

One of the main problems raised by the authors is data security and confidentiality. Long-term memory, regardless of whether it is implemented through summarization, RAG, or hierarchical systems, is essentially a permanent repository of users' personal data [18].

In general, the analysis confirms the actuality of this study and highlights the urgent need for a methodology that strategically combines the complementary strengths of both Sliding Window Context (for immediate coherence) and RAG (for long-term detail retention), while simultaneously integrating an advanced AI agent layer for cognitive control and state management.

3. Development of the Hybrid Method

The development of a robust and coherent LLM-based dialogue system requires hybrid architecture capable of overcoming the inherent limitations of conventional memory approaches. The primary goal is to create a methodology that systematically integrates long-term factual recall with immediate conversational coherence and dynamic state analysis [19].

Some part of architecture of Hybrid Method will bases on Sliding Window Context and RAG methods.

Since the hybrid methodology is based on combining the RAG and Sliding Window Context methods, their key stages were integrated. The enhancement involves fusing the strengths of both approaches and introducing new, dedicated stages for cognitive control [20].

The Sliding Window Context method involves the following key stages (Fig. 1):

International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 9 Issue 10, October – 2025, Pages: 7-18

Stage 1. Initialization: an empty contextual buffer of a fixed size *N* (e.g., 4000 tokens) is created.

Stage 2. Message Receipt: the system intercepts a new request from the user.

Stage 3. Context Update: the current contextual buffer is merged with the new request. If the total size exceeds *N* tokens, the beginning of the buffer is truncated to fit only the last *N* tokens.

Stage 4. Response Generation: the updated context, along with the query, is passed to the large language model for response generation.

Stage 5. Storage: the generated response is added to the contextual buffer.

Stage 6. Repetition: Stages 2-5 are repeated for each request.

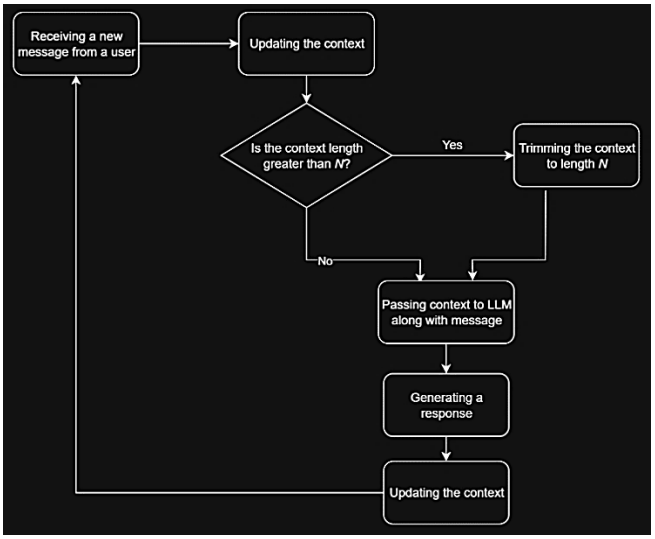

Fig. 1. *Algorithm of Sliding Window Context Method*

The RAG method involves the following stages (Fig. 2):

Stage 1. Initialization: a vector database is created for the future storage of the dialogue history.

Stage 2. Message Receipt: the system intercepts a new request from the user.

Stage 3. Vectorization: the user's query is converted into a vector representation (embedding) using an encoder model.

Stage 4. Search for Relevant Data: the obtained vector is used to search for the most similar records in the vector database.

Stage 5. Context Formulation: the initial user query is combined with the most relevant found records from the vector database.

Stage 6. Response Generation: the formulated context is passed to the large language model for response generation.

Stage 7. Database Update: the new query and the generated response are added to the vector database for future use.

Stage 8. Repetition: Stages 2-7 are repeated for each request.

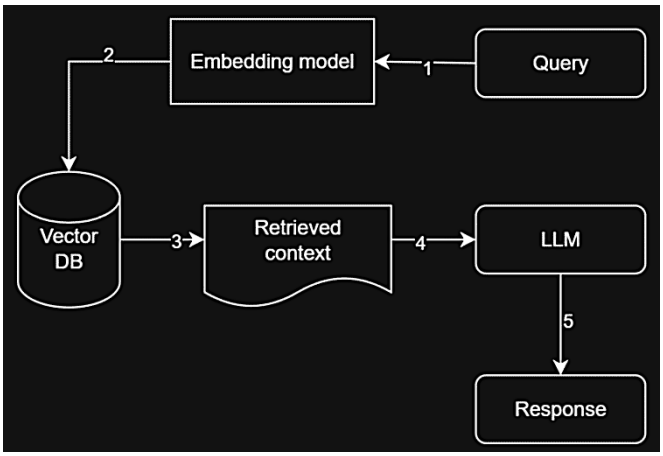

Fig. 2. *Algorithm of RAG Method*

Overall, the development of hybrid architecture consists of several steps.

Step 1: Development the Algorithm of Hybrid Method

Since the hybrid methodology is based on combining the RAG and Sliding Window Context methods, their key stages were integrated.

The enhancement involves fusing the strengths of both approaches and introducing new, dedicated stages for cognitive control.

To resolve the identified gaps in existing RAG and Sliding Window Context methods, a four-level memory architecture was designed [21].

The hybrid methodology is based on this four-level memory architecture, which synchronously processes and stores data to form the most complete context possible:

Level 1: Static Memory (Knowledge Base): this level stores constant data that forms the basis of the interaction. This includes bot profiles (e.g., role-playing characters or technical support specialists), general environment information (a fictional world or a corporate ecosystem), and initial metadata. This information is stored in a database and serves as the primary source of authoritative, relatively permanent knowledge.

Level 2: Dynamic Memory (RAG implementation): this level handles the storage and intelligent extraction of the entire dialogue history by executing the RAG method steps. Every chat message saved in the history is vectorized, and upon receiving a new query, the system performs a search to find semantically similar messages from the entire history.

Additionally, the system applies prioritization, where recent and emotionally significant events receive a higher score, making them more relevant for context inclusion.

Level 3: Contextual Analysis (AI Agent Layer): this level dynamically updates information regarding states, emotions, and environmental influence. After each message, a specialized AI agent is launched for analysis. This agent analyzes the interaction, identifies participants (if multiple users or bots are present), and, using the full context (profiles, history, existing states, and relationships), is able to understand hidden meanings and emotional subtext. Based on this analysis, database records concerning relationships between the bot's persona and the user, their emotional state, and the plot development are updated in real-time.

Level 4: Operational Memory (Context Sliding Window implementation): this level stores the most immediately relevant information for the response, implementing the steps of the Sliding Window Context method. The newest messages have the highest priority among the entire context, comprising the last 15-25 messages in chronological order, along with current emotions and active states. The LLM always has access to the freshest events, which is key to maintaining a smooth and natural conversation.

The schematic representation of the proposed hybrid approach is illustrated in Fig. 3.

The process of forming the prompt for the LLM is multi-layered: it begins with the most relevant data retrieved via semantic search (Level 2), is augmented with information from Static Memory (Level 1), Contextual Analysis (Level 3), and Operational Memory (Level 4). This structure ensures the completeness and multi-dimensionality of the context, forming the foundation for generating high-quality responses.

Step 2: Database Architecture Modeling

The implementation of the hybrid methodology, which is based on the four-level memory architecture, necessitates a dedicated database (DB) model whose structure accommodates all levels of contextual filling [22]. The DB architecture serves as the foundation for method implementation and must clearly demarcate different memory types to ensure their effective interaction.

The schematic representation of the DB structure for the hybrid method is shown in Fig. 4.

The Static Memory (Level 1) is governed by the "worldbuilding", "story_meta", and "characters" tables. These tables form the core of the constant data. A key feature is the inclusion of an embedding field for vectorization of this information, which enables the system to rapidly retrieve baseline knowledge via semantic search. The characters table, containing detailed profiles with their vector representations, is the nucleus of the static memory.

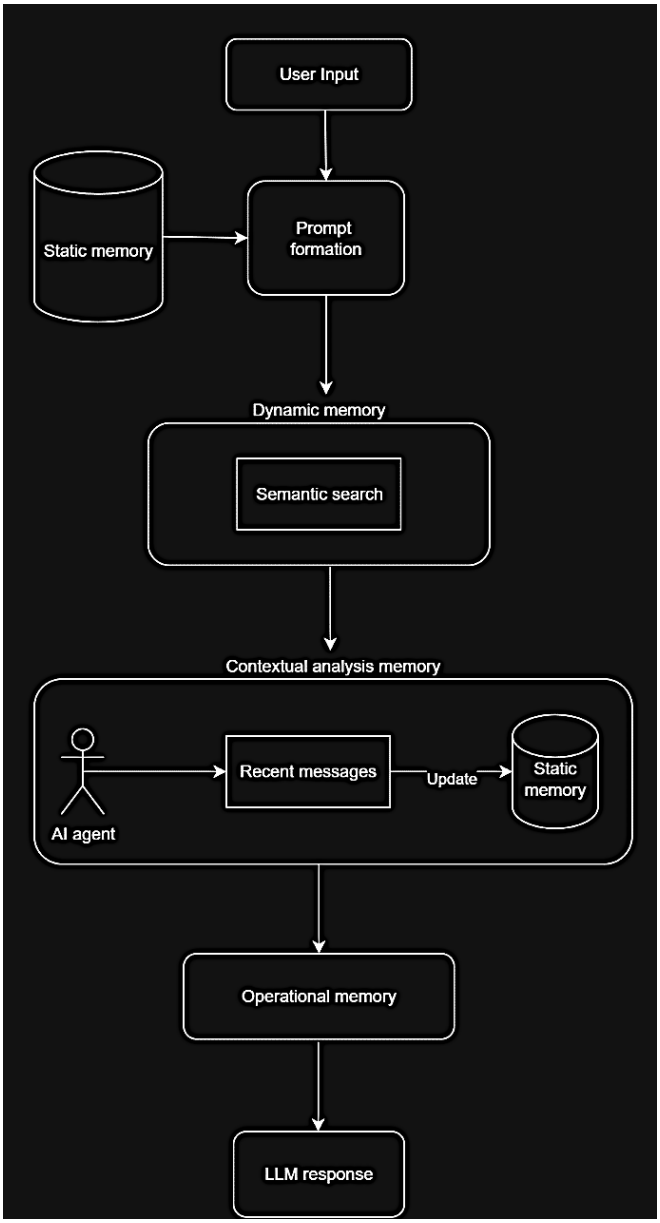

Fig. 3. *Algorithm of Hybrid Method*

The "story_log" table is the central component of the Dynamic Memory (Level 2). It stores the complete dialogue history, capturing every message. The critically important embedding field transforms this history into a vector space, enabling high-precision RAG. This table serves as the external, scalable memory, allowing the LLM to access any part of the conversation, regardless of its age.

Contextual Analysis (Level 3) is based on the "character_relationships" and "story_arcs" tables. These tables reflect the dynamic development of events and relationships within the chat. The "character_relationships" table stores the evolution of bonds between personas, tracking their type and intensity level.

International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 9 Issue 10, October – 2025, Pages: 7-18

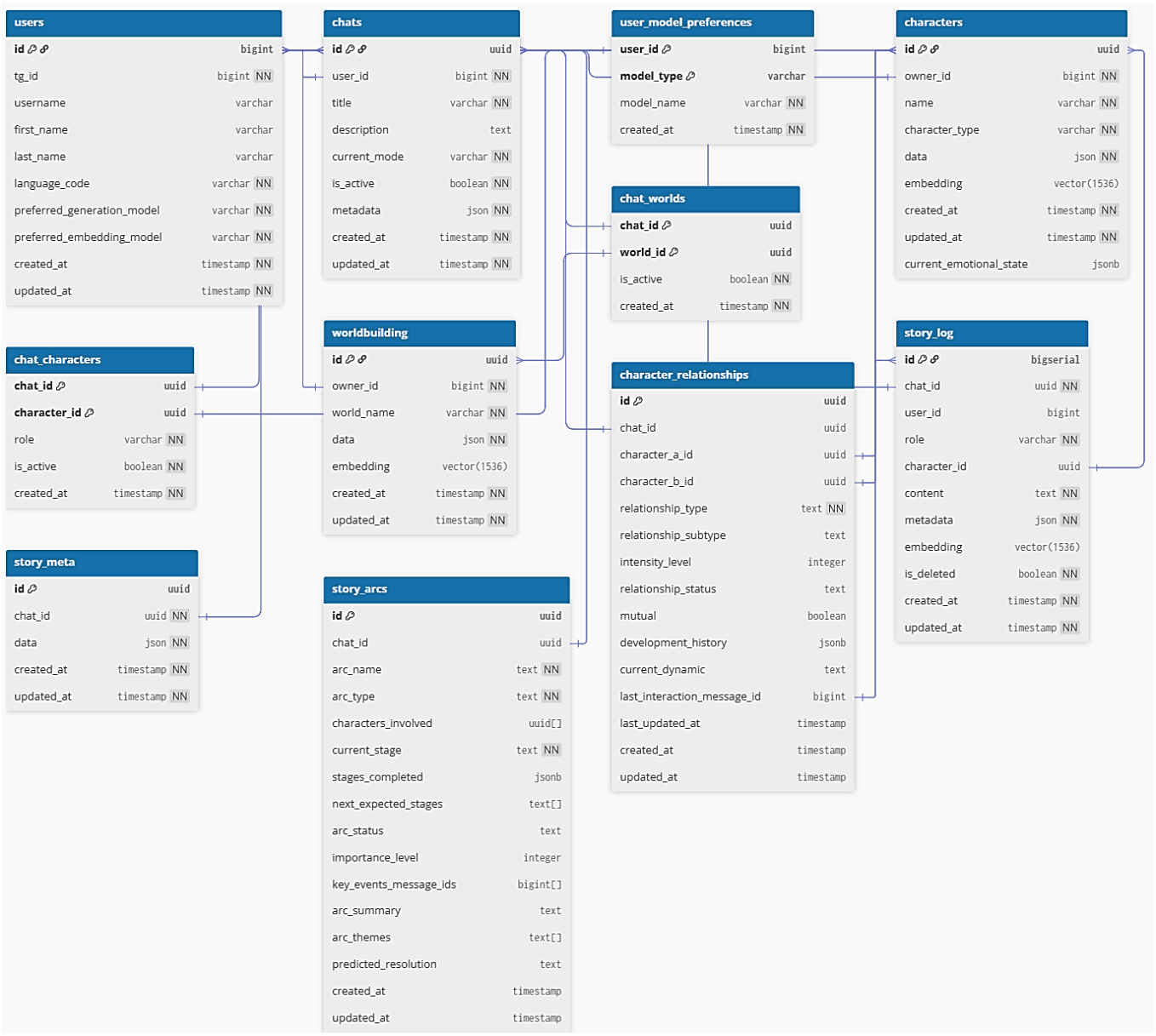

Fig. 4. *The schematic representation of the DB structure for the Hybrid Method*

The "story_arcs" table monitors active plot lines and their stages. These tables are the output of the AI Agent layer, which analyzes the dialogue and updates this "living" memory, enabling the system to sustain deep, coherent conversations.

Operational Memory (Level 4) is implemented via an optimized query to the "story_log" table. The system selects the last 15-25 messages, which receive the highest priority. This "sliding window" concept ensures immediate access to the most current context, essential for the naturalness and speed of the response.

Thus, the proposed DB schema is an optimal architectural solution that unifies static data, a complete dialogue history, dynamic relationship analysis, and operational access to recent messages. This allows the system not only to generate answers but also to support a multi-dimensional, realistic interaction [23].

Step 3: Instruments for Software Implementation of the Hybrid Method

The asynchronous framework **aiogram** (for the Telegram Bot API) was selected for the software implementation of the multi-component hybrid methodology application. This choice is justified by the requirements of the final platform, the necessity to minimize architectural complexity at the user interface level, and the need to ensure conditions for comprehensive testing.

International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 9 Issue 10, October – 2025, Pages: 7-18

The decision to use aiogram instead of a traditional web framework for the hybrid methodology is based on several architectural and application advantages:

- Interface Simplicity and Testing Focus: the application of aiogram allows the integration of complex business logic (four-level memory, RAG queries, relationship updates) without the necessity of developing a separate frontend (web interface). Since API documentation alone might be insufficient for convenient testing of a complex architecture, and frontend development is time-consuming and exhaustive within the scope of this research, using Telegram's native functionality (messages, buttons, and commands) for the entire user interface significantly reduces the overall project scope and complexity.

- Architectural Justification (Asynchronicity): the choice of aiogram, an asynchronous framework, is critically important due to the resource-intensive nature of the hybrid methodology (vectorization, semantic search in the DB, relationship analysis via the AI Agent). Asynchronicity enables efficient management of parallel input/output operations, such as network requests to the LLM or waiting for a response from the vector database, which substantially increases the chatbot's response speed.

This solution allows the main efforts to be concentrated at the business logic level, where the scientific novelty and practical value of the research are realized [24]. Furthermore, this approach enables direct evaluation of the hybrid methodology's effectiveness under conditions close to final use, which is more informative than testing separate API endpoints.

Thus, the selection of aiogram is an architecturally sound decision, providing the necessary balance between the complexity of the internal business logic and the simplicity of its final implementation and testing. An example of the entry point code for the aiogram chatbot is shown in Fig. 5 [25].

Step 4: Software Implementation of the Hybrid Method

The software realization of the hybrid methodology is based on a multi-layered, asynchronous system where memory is strictly separated into four dynamically interacting levels. This structure ensures high throughput and the ability to process multiple context layers in parallel [25].

The "ContextRetriever" class is the central component that implements the logic for uniting all memory levels. The system acknowledges that different types of memory have different importance, which is implemented through smart thresholds. The "retrieve_context" method initiates a hybrid search, using semantic search (via vectors) for Dynamic Memory and a full selection for Static Memory. This function coordinates the query vectorization and delegates the hybrid search to the "VectorStore".

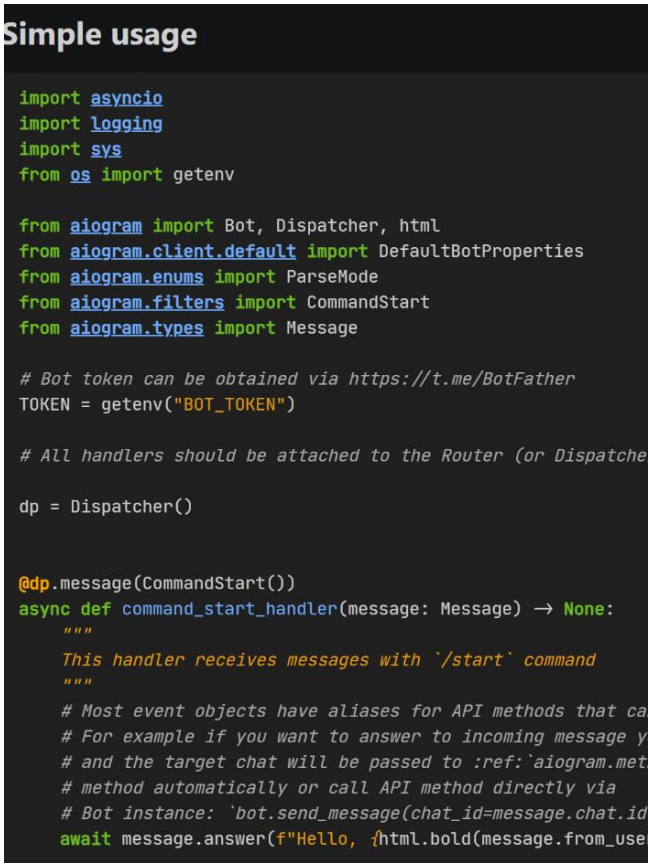

Fig. 5. *The Code example of aiogram framework from official documentation*

The dynamic relationships between chatbot personas are implemented via the "retrieve_character_relationships" method. This programmatic mechanism ensures that the context always includes the current state of interactions. The logic is encapsulated within the "RelationshipManager" (responsible for manipulation with the "character_relationships" table), handling complex SQL logic and processing JSONB fields ("current_dynamic", "development_history").

To optimize the context window, the "get_relationship_summary_for_character" method is used. This function performs a compression of the relationship graph into a single, concise textual description, thereby reducing the burden on the final prompt.

A key programming technique is the assignment of a synthetic, high "similarity_score"=18.0 to these critical state elements (as well as story arcs and emotional states). This value significantly exceeds the typical range (≤1.0 for standard vector search) and guarantees that this element will always be among the highest-priority results, regardless of the semantic similarity of the user's current query.

International Journal of Academic Information Systems Research (IJAISR)
ISSN: 2643-9026
Vol. 9 Issue 10, October – 2025, Pages: 7-18

Analogous functions ("retrieve_story_arcs" and "retrieve_emotional_state") extract and prioritize structured data in the same manner.

The "RelationshipAnalysisAgent" class provides the software implementation for automatic learning and updating of dynamic elements (Level 3). The agent is initiated after every dialogue exchange (user message + AI response). Its objective is to determine how the interaction impacted the "character_relationships" and "current_emotional_state".

The agent does not merely analyze the latest message; it calls "_build_full_context", gaining access to all memory layers (RAG, relationships, and arcs). This allows the LLM-agent to make informed decisions about relationship changes, understanding the overall plot and emotional background.

The final step, "_apply_relationship_updates", saves the changes, effectively providing self-updating contextual memory. This guarantees that the memory is dynamic and evolves with every new conversational step.

The software architecture of the hybrid methodology is implemented through a multi-module system of classes that ensure the asynchronous operation of the four memory levels.

The Class Diagrams of Core logic and RAG implementing are shown in Fig. 6 and Fig. 7.

The Core Logic Class Diagram describes the architecture for managing static, dynamic, and contextual memory. The RAG Class Diagram describes the components responsible for vectorization and searching for relevant fragments.

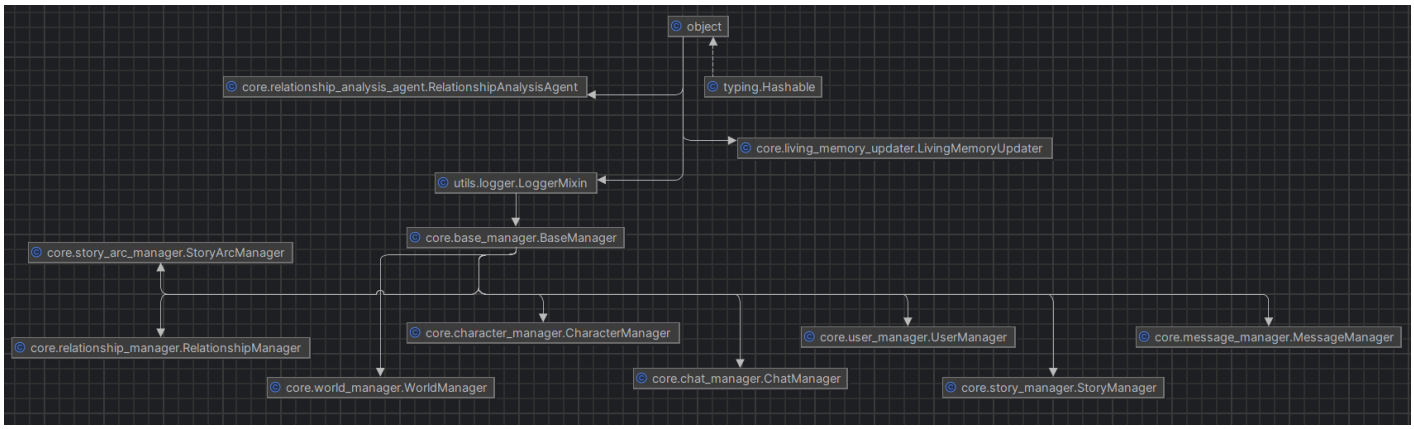

Fig. 6. *The Class Diagram of Core Logic*

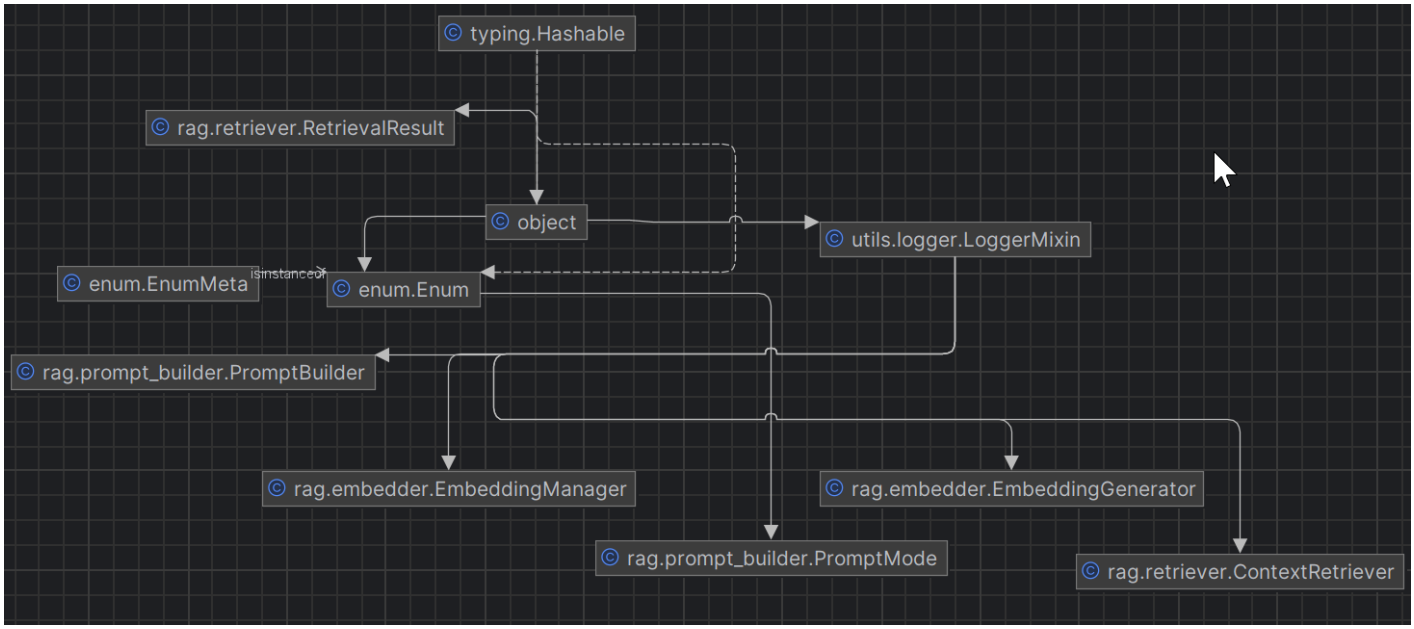

Fig. 7. *The Class Diagram of RAG Logic*

4. The Testing

For thorough test coverage, test data will be generated based on a specific role-playing scenario designed to demonstrate the quality and efficacy of the developed hybrid memory enhancement methodology. The testing must be transparent; therefore, a simple and linear scenario will be created that requires the logical recall of facts introduced at the very beginning of the chat [26].

The test scenario is structured to challenge the long-term memory capabilities of the LLM by creating a significant gap between the initial introduction of critical facts and their necessity for a logical conclusion, thereby validating the operation of the four-level hybrid memory architecture.

Tables 1-4 describe the scenario, its key aspects, and the expected testing outcomes based on coherence criteria.

Testing will be considered successful if the language model in its generated response to the facts from the third stage recalls, logically connects, and draws conclusions based on the key facts from the first stage. Facts from Stage 3 and Expected Results are shown in Table 5.

Table 1: Aspects of the Test Scenario

Aspect	Description	Function
Plot	19th century, England. An Investigator travels in a carriage to investigate a fire at Lord Carroll's mansion and the theft of the family Talisman (which is actually a treasure map).	Creation of data for testing the hybrid method's memory enhancement capabilities.
Scenario Stages	Stage 1. Travel with a coachman who has unique features. Stage 2. Independent investigation that should "shift" the LLM's focus and "lose" the facts from the first stage. Stage 3. A final twist, where a witness provides a description of the thief that matches the coachman.	Verification of the hybrid method's ability to link long-forgotten details from the beginning of the chat (coachman's features) with critical evidence at the end (thief's description) after a significant dialogue interval.
Key Link (Expected Result)	The LLM will receive information about the coachman's features and must use it in its reaction when the witness mentions the same details at the end of the scenario.	Verification of the hybrid method against the expected result.

Table 2: Role Personas and Scenario Role

Persona	Goal in the Scenario	Role
Investigator	Collection of facts at the start and the recalling of facts with logical conclusions by the LLM at the end.	Stages 1, 2 – User. Stage 3 – LLM.
Coachman	Introduction of key facts.	User.
Witness (Lord Carroll's housekeeper)	Hinting at the key facts mentioned at the start of the scenario at the end.	User.
Lord Carroll (Victim)	Background persona for fact gathering.	Mentioned in the chat.

Table 3: Key Facts and Logical Link

Key Fact	Witness Mention	Link
The coachman has an eye patch.	The culprit has heterochromia (different colored eyes).	The coachman wears an eye patch to hide a key physical detail.
The coachman speaks German when emotional.	The culprit shouted in German when the witness saw them during the crime.	Coincidence of traits.
The culprit knew the room layout and the Talisman's secret.	The Lord rode in the coachman's carriage two weeks ago while intoxicated, and the coachman helped carry him home.	The coachman learned about the Talisman from the Lord while he was incoherent and saw the room layout when helping him to the manor.

Table 4: Testing Stages and Message Range

№	Message Range	Description
1	{1-6}	Travel with the coachman who has unique features. The model receives key information.
2	{7-59}	Independent investigation, which should "shift" the language model's focus and "lose" the facts from the first stage.
3	{60-63}	Final twist, where the witness gives a description of the thief that matches the coachman. The model must "recall" and "cross-reference" facts from the first stage.

Table 5: Role Personas and Scenario Role

Fact	LLM Reaction (Expected Result)
The culprit has heterochromia.	LLM "recalls" the coachman's eye patch. LLM "cross-references" the eye patch and heterochromia.
The culprit speaks German when emotional.	LLM "recalls" the coachman shouted in German. LLM "cross-references" the coincidence.
The culprit knew the room layout and the Talisman's secret.	LLM "takes into account" the Lord's intoxicated state and the possibility of revealing the secret to the coachman. LLM "cross-references" that if the coachman helped the Lord to the manor, he saw the room layout.
The Lord rode in the carriage home 2 weeks ago.	LLM concludes that it was the same coachman.

If the LLM's generated response recalls and cross-references the listed logical conclusions, the long-term memory test for the **Hybrid Methodology** will be passed.

Since the test set is comprised of messages, a set of test messages in JSON format was created.

The set includes 63 messages spanning all test stages (as outlined in Table 4). The effectiveness based on the expected results will be evaluated in the key LLM-generated response. The semantic content of this response will be assessed against the expected coherence criteria [27].

Fig. 8 provides a screenshot of the chatbot application showing the LLM-generated response at the final critical stage (Stage 3).

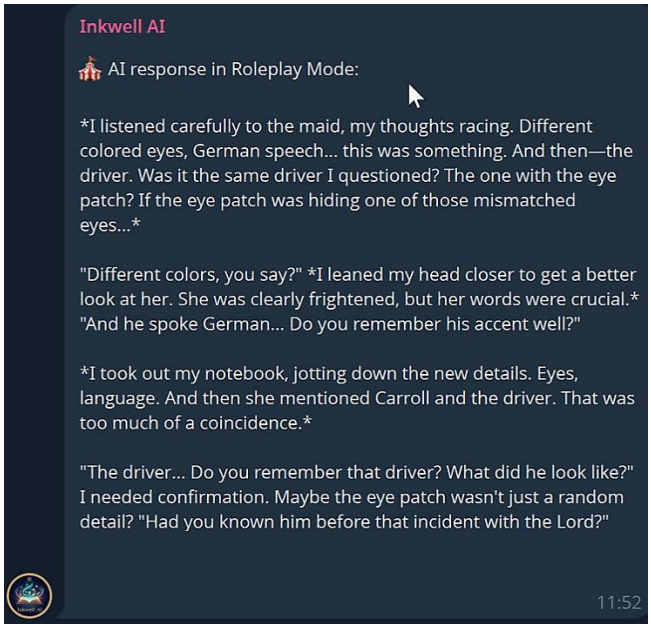

Fig. 8. *The screenshot of the chatbot application showing the LLM-generated response*

The generated LLM response clearly demonstrates the model's ability to recall and cross-reference multiple facts that were introduced significantly earlier in the dialogue (messages 1-6), validating the system's long-term memory. The response explicitly contains the expected logical connections:

- *"Different colored eyes, German speech... this was something. And then—the driver. Was it the same driver I questioned? The one with the eye patch? If the eye patch was hiding one of those mismatched eyes..."* (Recalls the eye patch and connects it to heterochromia).
- *"And he spoke German... Do you remember his accent well?"* (Recalls the German speech and asks for confirmation).
- *"And then she mentioned Carroll and the driver. That was too much of a coincidence."* (Draws a direct logical conclusion about the Lord's connection to the coachman).

The received results were rigorously analyzed against the success criteria defined in Table 6.

Table 6: The analysis of received results

Fact	LLM Reaction (Expected Result)	Response Analysis	Status
The culprit has heterochromia	LLM "recalls" the coachman's eye patch. LLM "cross-references" the eye patch	LLM explicitly recalled the fact about the eye patch in its internal monologue: "The one with the eye patch?"	Passed
The culprit speaks German when emotional	LLM "recalls" the coachman shouted in German. LLM "cross-references" the coincidence	LLM recalled German speech and cross-referenced it with the coachman in the monologue	Passed
The culprit knew the room layout and the Talisman's secret	LLM "takes into account" the Lord's intoxicated state and the possibility of revealing the secret to the coachman. LLM "cross-references" that if the coachman helped the Lord to the manor, he saw the room layout	LLM concluded the excessive coincidence, focusing on the driver's connection to Carroll but LLM did not make a direct explicit conclusion but emphasized the coincidence of the driver with the one who helped the Lord	Passed
The Lord rode in the carriage home 2 weeks ago	LLM concludes that it was the same coachman	LLM directly questioned the coachman's identity and nature: "The driver... Do you remember that driver?"	Passed

The hybrid method successfully passed the long-term memory test against all critical criteria.

The model not only recalled all three key, long-forgotten facts (the eye patch, the German language, and the connection to the Lord's return) but also logically cross-referenced them in its internal monologue and used these connections to formulate critical next questions [28].

The direct conclusion about the room layout was covered by the overall deduction of "too much of a coincidence," which is sufficient for high-level reasoning.

This confirms that for complex dialogue systems requiring logical circumstances and detailed accuracy, the hybrid approach is a highly effective solution.

5. Conclusion

The results are confirmed that hybrid architecture is the most effective strategy for ensuring long-term contextual coherence in complex dialogue scenarios.

The central result of this work is the development and empirical substantiation of a novel hybrid methodology that integrates the strengths of RAG, Sliding Window Context, and a specialized AI Agent layer.

The proposed four-level memory architecture (Static, Dynamic, Contextual Analysis, and Operational) successfully addressed the key limitations of isolated methods:

- The irreversible loss of information inherent in the Sliding Window Context was completely nullified by integrating RAG (Dynamic Memory), which ensures the permanent storage and retrieval of the entire dialogue history.
- The issue of lack of cognitive control and the generation of inconsistent, "mechanical" responses in traditional RAG was overcome by the Contextual Analysis (AI Agent) layer, which dynamically tracks states, relationships, and emotional context.

The effectiveness of this hybrid approach was confirmed through testing a complex role-playing scenario involving a long dialogue interval (63 messages) designed to challenge long-term memory.

The testing demonstrated that the LLM, powered by the hybrid method, successfully recalled, logically cross-referenced, and utilized multiple facts introduced early in the conversation (messages 1-6) for a critical deduction at the end (messages 60-63).

Specifically, the model achieved a high success rate by correctly linking three keys, previously "forgotten" facts (the eye patch, German speech, and the Lord's connection to the coachman) to form a coherent conclusion.

The scientific novelty of the work lies in developing this hybrid methodology that systematically accounts for the cognitive needs of an LLM through the strategic combination

of opposing memory approaches and the integration of a dynamic, self-updating AI agent layer [29].

The successful results confirm that the strategy of integrating specialized AI agents into the process of context formation and retrieval is highly promising. Further enhancement of the methodology will focus on scaling the agent-based logic:

- Context-Dependent Fact Weighting: developing the AI Agent's logic to implement algorithms that assign increased significance (weight or tags) to unique or critical facts (e.g., visual markers or plot-defining events). This would guarantee their inclusion and prioritized position during prompt formation, even if their semantic similarity is low.

- Multiple Vector Generation: moving beyond a unified vector representation for each message. The agent could be empowered to generate multiple vectors for key entities, events, or conflicts within a single message. This alternative pathway would minimize the risk of semantic loss and substantially increase the precision of relevant fragment selection by the RAG system.

- Active Recall Mechanism: to mitigate the risk of LLM potentially ignoring retrieved facts (even when available in the prompt), a mechanism for Active Recall is proposed. This intermediate module would analyze the LLM's generated response for any critical memory facts that were extracted but ignored. In such cases, the module would interpolate the input context with an additional "reminder" to the LLM, stimulating the necessary logical integration (e.g., forcing a comparison of the extracted facts).

These directions for scaling the AI agents are promising for developing an active cognitive system that will significantly enhance the depth and authenticity of the user's dialogue experience.

6. Acknowledgment

The papers acknowledge the support of the Department of Informatics, Kharkiv National University of Radio Electronics, Ukraine, in numerous help and support to complete this paper.

The research results were obtained as part of the EU Horizon Europe international research project INITIATE (grant No. 101136775).

7. References

[1] Xiong, Z. et al. (2025). How memory management impacts LLM agents: An empirical study of experience-following behavior. [Online]. Available: https://arxiv.org/abs/2505.16067.

[2] Zeng, R., Fang, J., Liu, S., & Meng, Z. (2024). On the structural memory of LLM agents. [Online]. Available: https://arxiv.org/abs/2412.15266.

[3] Liu, P. et al. (2023). A Survey of large language models. [Online]. Available: https://arxiv.org/abs/2303.18223.

[4] Lewis, P. et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. [Online]. Available: https://arxiv.org/abs/2005.11401.

[5] Karakonstantyn, D., & Tvoroshenko, I. (2024). About the issue of optimization the performance of the server part of the information system, in Proc. XII Int. Sci. Pract. Conf. "Prospective directions of modern science and education in the world", Rotterdam, International Science Group, Netherlands, pp. 364-366.

[6] Wang, J. et al. (2023). A survey of large language models for dialogue systems. [Online]. Available: https://arxiv.org/abs/2311.16789.

[7] Zhang, Y., Yu, Z., Jiang, W., Shen, Y., & Li, J. (2023). Long-term memory for large language models through topic-based vector database, in Proc. 2023 Int. Conf. Asian Lang. Process. (IALP), United States, pp. 258-264.

[8] Gu, J. et al. (2024). Long-context LLMs: A comprehensive survey. [Online]. Available: https://arxiv.org/abs/2404.09886.

[9] Wang, Q., Fu, Y., Cao, Y., Wang, S., Tian, Z., & Ding, L. (2023). Recursively summarizing enables long-term dialogue memory in large language models. [Online]. Available: https://arxiv.org/abs/2308.15022.

[10] Gao, Y. et al. (2024). Retrieval-augmented generation for large language models: A survey. [Online]. Available: https://arxiv.org/abs/2312.10997.

[11] Humanloop. (2024). RAG architectures: An overview of retrieval-augmented generation architectures. [Online]. Available: https://humanloop.com/blog/rag-architectures.

[12] Zhao, Y., Cao, H., Zhao, X., & Ou, Z. (2024). An empirical study of retrieval augmented generation with chain-of-thought, in Proc. 2024 14th Int. Symp. Chin. Spok. Lang. Process. (ISCSLP), pp. 436-440.

[13] Wang, Z., Teo, S. X. M., Ouyang, J., Xu, Y., & Shi, W. (2024). M-RAG: Reinforcing large language model performance through retrieval-augmented generation with multiple partitions. [Online]. Available: https://arxiv.org/abs/2405.16420.

[14] Zhang, Y., Wu, H., Xu, Z., Gu, C., Ma, S., & Li, M. (2024). Evaluating very long-term conversational memory of LLM agents. [Online]. Available: https://arxiv.org/abs/2402.17753.

[15] Karunathilake, K. A. G. G., Abhayawardhana, C. G., De Silva, M. I. W. R., & Gamage, S. N. (2023). The right to be forgotten in the era of large language models: implications, challenges, and solutions. [Online]. Available: https://arxiv.org/abs/2307.03941.

[16] Tvoroshenko, I., Pomazan, V., Gorokhovatskyi, V., & Kobylin, O. (2023). Application of video data classification models using convolutional neural networks, Int. J. Acad. Appl. Res., vol. 7, no. 11, pp. 134-145.

[17] Tvoroshenko, I., Gorokhovatskyi, V., Kobylin, O., & Tvoroshenko, A. (2023). Application of deep learning methods for recognizing and classifying culinary dishes in images, Int. J. Acad. Appl. Res., vol. 7, no. 9, pp. 57-70.

[18] Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks, Int. J. Acad. Inf. Syst. Res., vol. 7, no. 7, pp. 25-36.

[19] Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Handwritten character recognition models based on convolutional neural networks, Int. J. Acad. Eng. Res., vol. 7, no. 9, pp. 64-72.

[20] Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., Hudáková, M., & Gorokhovatskyi, O. (2024). Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, IEEE Access, vol. 12, pp. 73376-73385.

[21] Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., & Hudáková, M. (2025). Image description compression in classification structural methods, IEEE Access, vol. 13, pp. 43631-43641.

[22] Gorokhovatskyi, V., Tvoroshenko, I. & Yakovleva, O. (2024). Transforming image descriptions as a set of descriptors to construct classification features, Indones. J. Elec. Eng. Comput. Sci., vol. 33, no. 1, pp. 113-125.

[23] Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2023). Statistical data analysis models for determining the relevance of structural image descriptions, IEEE Access, vol. 11, pp. 126938-126949.

[24] Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Tools for fast metric data search in structural methods for image classification, IEEE Access, vol. 10, pp. 124738-124746.

[25] Aiogram 3.22.0 documentation. (2025). Simple usage. [Online]. Available: https://docs.aiogram.dev/en/dev-3.x/.

[26] Tvoroshenko, I., & Gorokhovatskyi, V. (2022). The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, Int. J. Eng. Inf. Syst., vol. 6, no. 2, pp. 40-48.

[27] Ayaz, A. M., Gorokhovatskyi, V., Tvoroshenko, I., Vlasenko, N., & Khalid, M. S. (2021). The research of image classification methods based on the introducing cluster representation parameters for the structural description, Int. J. Eng. Trends Technol., vol. 69, no. 10, pp. 186-192.

[28] Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Al-Dhaifallah, M. (2022). Classification of images based on a system of hierarchical features, Comput. Mater. Contin., vol. 72, no. 1, pp. 1785-1797.

[29] Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods, Adv. Inf. Syst., vol. 9, no. 1, pp. 5-12.