Model and methodology for the formation of intelligent human factor error management with adaptive self-learning in large language models*

Yaroslava Momryk[1,†] and Dmytro Sabodashko[1,*,†]

[1] Lviv Polytechnic National University, 12 Stepan Bandera str., 79013 Lviv, Ukraine

Abstract

This paper analyzes and proposes an approach to the self-learning of artificial intelligence systems based on Large Language Models (LLMs) for handling human factor errors using human-like logic. The concept of the "human factor in AI" is considered as a dual process: on the one hand, the documentation and systematization of user errors, and on the other hand, the application of human reasoning to improve system responses and to formulate correction rules.

The operating principle is based on a knowledge base methodology: the Human Error Log accumulates data with the possibility of user confirmation of proposed corrections; the collected statistics are generalized and transformed into new processing rules, which must obligatorily undergo human verification. The necessity of this mechanism is justified as a key means of protection against incorrect self-learning and adaptation to harmful or manipulative behavior.

The proposed architecture includes the following modules: Input Analyzer, Keyboard/Layout Detector, Human Error Log, Threshold Engine, Human Verification Layer, and Model Adapter, which together form a multi-level control system. The optimality of the approach lies in the application of an interface-level correction block: the core of the large language model remains universal, while local rules are implemented at the pre-processing pipeline level, ensuring resistance to manipulation and overall system stability.

Human verification guarantees the transparency of algorithmic changes, compliance with the principles of human-centered AI, and the practical feasibility of scalable and safe adaptation of the system to real-world user errors.

Keywords

human-centered artificial intelligence, safe and controlled self-learning, human-in-the-loop verification, knowledge base adaptive learning, knowledge base validation, trustworthy AI handling human factor errors, system architecture, information security, multi-level security in Large Language Models.

1. Introduction

Modern intelligent text processing and human-computer interaction systems are rapidly evolving; however, most of them are designed primarily according to machine logic. Data correction is typically performed based on general models or simple rules, without considering individual user errors. A Large Language Model (LLM) is a type of artificial intelligence trained on massive amounts of textual data to understand, generate, and process human language. LLMs can perform tasks such as translation, text generation, information summarization, and dialogue management by leveraging statistical patterns in language.

These models, however, face the challenge of the human factor — repetitive errors caused by cognitive characteristics, fatigue, or interface limitations (e. g., incorrect keyboard layout or casing). Such errors are rarely used as signals for system adaptation.

Over the past decade, there has been a significant shift from technically-oriented AI models toward systems that account for cognitive, behavioral, and contextual human characteristics. This approach, known as Human-Centered Artificial Intelligence (HCAI) , has become the foundation for developing intelligent systems that not only respond to user input but also adapt to individual habits, errors, and behavioral patterns.

Despite this progress, most contemporary language models and digital services remain largely insensitive to specific human errors, such as incorrect keyboard layout, unintentional casing

* CSDP'2026: Cyber Security and Data Protection, May 7, 2026, Lviv, Ukraine

* Corresponding author.

† These authors contributed equally.

yaroslava.b.momryk@lpnu.ua (Y. Momryk); sabodashko.dv@gmail.com (D. Sabodashko)

0009-0001-4114-0347 (Y. Momryk); 0000-0003-1675-0976 (D. Sabodashko)

changes, or random micro-errors caused by fatigue or fast typing. These errors are generally treated as “noise” or “anomalies” rather than as valuable data for system adaptation.

Consequently, a gap exists between the capabilities of modern models and the real behavioral patterns of users. Investigating this gap is particularly important given the growing use of LLMs in various security-critical domains, including confidential information detection [27], compliance assessment [28], and software vulnerability analysis [29]. It is also crucial in the context of building self-learning services capable of accounting for the human factor while maintaining control over system quality and safety.

Human-Centered Artificial Intelligence (HCAI) provides an approach that considers human cognitive and behavioral aspects and ensures oversight over automated system changes. This forms the scientific basis for integrating a Human Verification Layer (HVL) into self-learning systems, allowing the system to propose changes that are subsequently confirmed by a human.

Motivation for a Human-Centered Self-Learning Approach.

AI self-learning is already widely applied to reduce the impact of human errors in everyday tools, ranging from search engines and text editors to programming environments and online shopping platforms. These mechanisms enhance productivity and decrease the risk of incorrect outputs.

One common manifestation of the human factor is the accidental entry of text using an incorrect keyboard layout. For example, a user may type “gjnhs,yj” instead of the intended word “потрібно” (“necessary”). While this error is immediately recognizable to a human, for a computer it is merely a sequence of characters. AI self-learning enables the system to identify such patterns and suggest the correct input.

Similar mechanisms have already been implemented in search engines (e.g., Google Search’s “Did you mean…” suggestions), text editors (e.g., Grammarly’s spelling and grammar corrections), integrated development environments (IDEs) for programmers (e.g., automatic syntax correction), and smartphone autocorrection services (see Table 1). In each case, the algorithm learns from the user’s input history and adapts to individual habits, demonstrating the potential of adaptive, error-aware AI systems.

The problem is that most existing methods of protection or threat detection do not account for the real-time behavioral dynamics of traffic flows, do not adapt to new abnormal influences, and do not perform a comprehensive analysis of the traffic generation environment [1, 7]. Therefore, there is a need to develop a method capable of detecting hidden, unauthorized influences through intelligent analysis of network interactions, using complex anomaly models, behavioral profiles, and mechanisms to explain the decisions made.

Table 1
Examples of Error-Correcting Systems

System/Service	Type of Errors Corrected	How Self-Learning Works	Benefits for the User
Google Search	Spelling and typographical errors in search queries	Analyzes billions of queries and learns from the most frequent variants	Provides relevant results even for incorrect input
Smartphone Autocorrection (iOS, Android)	Typing errors, missing letters, incorrect keyboard layout	Remembers individual user typing habits	Faster typing with fewer manual corrections
Grammarly / LanguageTool	Grammar, style, and spelling errors	Uses NLP models trained on large text corpora	Improves writing quality and adapts to the author’s style
IDE (Visual Studio Code, JetBrains)	Syntax errors, incorrect variable names	Learns from code patterns and libraries	Reduces bugs and accelerates programming
Google Translate / DeepL	Word input errors in a different language or with typos	Learns from parallel text corpora	Increases translation accuracy even with incorrect input
Gmail Smart Compose / Outlook	Incomplete phrases, typing errors in emails	Learns from millions of email examples	Provides relevant autocomplete suggestions, saving time
Amazon / eBay Search	Incorrect product names (e.g., “iphon” instead of “iPhone”)	Learns from search history	Helps find the correct product even with input errors

All of these systems demonstrate that AI self-learning can effectively reduce the impact of human errors, ranging from typographical to syntactic mistakes. However, a critical balance must be maintained: users should have the ability to confirm or reject corrections to prevent overcorrection. In cases such as errors caused by an incorrect keyboard layout, human verification is essential, although it is often implemented in a hidden or simplified form.

Thus, AI self-learning helps mitigate minor human factor errors-ranging from typing mistakes in search queries to syntax errors in programming. Similar to how Google Search corrects queries and Grammarly adapts to the author's writing style, algorithms can learn from a user's input history and automatically suggest correct alternatives. This reduces cognitive load, saves time, and smooths workflow. At the same time, it is critically important to allow the user to accept or reject corrections, preventing overcorrection or incorrect replacements. Consequently, keyboard layout error correction represents another example of AI self-learning benefits, illustrating a general trend: autonomous systems can compensate for human errors but require human verification for safe use.

However, LLM-based AI systems typically do not correct such errors automatically, as their role is often assumed to be preserving accuracy and keeping control in the user's hands. Nonetheless, implementing correction mechanisms, similar to a layout-aware autocorrect, with human verification would be beneficial. Such a feature improves the service by:

- Reducing cognitive load: Users do not need to manually correct each error; the system handles it automatically, saving time and effort.
- Enhancing adaptability: Self-learning enables the system to adjust to individual user habits (e.g., frequent switching between Ukrainian and English keyboard layouts).
- Increasing productivity: Fewer interruptions, faster text input, and reduced errors in documents.
- Expanding accessibility: Such functions are particularly useful for individuals with motor or cognitive difficulties, who are more prone to errors.

Therefore, analyzing and testing approaches to LLM-based AI self-learning as a tool for compensating human errors is highly relevant.

Research Aim and Objectives. The aim of this study is to substantiate the need for the systematization and analysis of errors in human-AI interactions, specifically those classified under the human factor, and to propose a solution for their classification and processing using a self-learning algorithm that incorporates human logic and psychological principles.

To achieve this aim, the study addresses the following key objectives:

1. Analyze common interaction scenarios and provide examples of errors that can be classified as human factor errors, highlighting instances where AI fails to detect or handle them.
2. Investigate current automated error detection tools and identify their limitations in terms of processing and adaptability.
3. Develop a conceptual model for integrating error detection, storage, and self-learning processes.
4. Formulate practical recommendations for the safe implementation of new rules, particularly to prevent incorrect or unsafe self-learning.

2. Literature review

Research on the human factor in artificial intelligence interactions has a long-standing history, originating from foundational studies in cognitive psychology and complex system safety.

Reason [1], in his seminal work Human Error, proposed a classification of errors that remains relevant today. He distinguished lapses, mistakes in action, and cognitive failures, which explain typical user problems when interacting with AI algorithms, such as incorrect data entry or misinterpretation of system outputs.

Dekker [2], in The Field Guide to Understanding Human Error, further developed this topic, emphasizing the systemic nature of errors. He demonstrated that mistakes often arise not from individual shortcomings but from organizational conditions, such as stress, haste, or lack of instructions. This perspective is particularly critical for AI applications in high-stakes domains, such as medicine, aviation, and transportation, where the human factor can amplify risks even in

the presence of automated systems.

Amershi et al. [3], in Guidelines for Human-AI Interaction, proposed a set of principles for designing interfaces aimed at minimizing the risk of user errors caused by incorrect parameter entry or misinterpretation of system prompts. Ergonomic interface design is highlighted as a key instrument for reducing the impact of the human factor.

Li et al. [4], in their survey Human Factors in AI Safety: A Survey, systematically reviewed research on the influence of human errors on AI system safety. They emphasize that even the most advanced algorithms can become hazardous if used incorrectly by humans.

Glikson and Woolley [5], in their article Human Trust in Artificial Intelligence: Review of Empirical Research, explore the psychological aspects of human-AI interaction. They show that trust in AI can be either excessive or insufficient. In the former case, users may blindly rely on algorithms; in the latter, they may ignore useful recommendations. Both scenarios lead to errors, highlighting the importance of considering cognitive biases and psychological factors.

Contemporary literature identifies several key research directions on the human factor in AI usage: theoretical models of errors [1, 2, 6, 7], practical approaches to risk mitigation [3], safety considerations in AI deployment [4], and psychological and cognitive factors [5]. These sources provide a foundation for further analysis and the development of strategies to reduce the impact of the human factor in AI interactions.

Key Research Directions. Human-Centered AI and Adaptive System Principles. Human-Centered Artificial Intelligence (HCAI) approaches focus on transparency and explainability of AI actions, fostering user trust, and integrating human oversight into the learning and adaptation process. Amershi et al. [3] propose clear guidelines for implementing human-in-the-loop mechanisms in adaptive AI systems: the system should inform users about changes, present alternative options, and leave the final decision to the human. These principles are particularly critical for services that learn from user behavior, such as text error correction systems.

Li and Gu [6] present a risk framework for human-centered AI, where human verification serves as a critical barrier against uncontrolled self-learning. Methods such as Delphi-AHP can be used to prioritize areas of control and ensure safe adaptation.

Human Factor and Error Classification. James Reason's Swiss Cheese model [1] classifies errors as slips (correct intention but incorrect execution, e.g., typing with the wrong keyboard layout), lapses (errors caused by inattention), and mistakes (errors in planning or strategy selection). These categories provide a structure for maintaining a Human Error Log (HEL) within AI systems, enabling the tracking of recurrent errors and the establishment of threshold rules for automated adaptation suggestions.

Reason [7] also emphasizes that humans are not only sources of errors but often prevent critical system failures. This highlights the importance of integrating a Human Verification Layer (HVL) into self-learning systems. Hollnagel [8] extends this view, proposing that system evaluation should consider not only errors but also daily successful operations, which require human verification.

Norman [9] and Shneiderman [10] argue that repeated user errors should be treated as valuable signals for improving system interaction rather than as mere "noise."

Input Error Correction and NLP Technologies. Studies by Caruana [11], Belinkov & Bisk [12], and Golding & Roth [13] demonstrate that natural language processing (NLP) systems are capable of identifying and correcting input errors at both the character and word levels. A notable subfield is keyboard layout error correction (KLE). Balyan et al. [14] show that language models can effectively detect text entered with the wrong keyboard layout and propose automatic corrections.

However, in current systems, correction is typically handled at the level of individual modules. There is no accumulation of user error statistics, nor is there human verification of proposed changes prior to their integration. This gap presents an opportunity for scientific innovation: combining adaptive NLP-based correction with human verification and an audit trail.

Self-Learning Systems and Continual Learning. The problem of catastrophic forgetting in neural networks during continual learning has been described by Parisi et al. [15] and Delange et al. [16]. Models that learn "on the fly" risk losing previously acquired knowledge or altering behavior in unpredictable ways. Chen & Liu [17] discuss this issue in the context of lifelong machine learning.

A safe solution is the adaptation of the pre-processing pipeline: a module for collecting human errors, a rule suggestion module, and an integration module activated only after user verification. This separation allows the system to learn without modifying the core model, reducing associated

risks. Zhang et al. [18] evaluate how AI can perform self-learning in a training context and emphasize the need for human verification of results.

Industrial Practices. Technical solutions such as Google's Gboard and Google Search employ advanced algorithms, including finite-state transducer (FST) decoders, multilingual support, transliteration, and noisy text correction. However, these systems do not generate rules based on large-scale repetitions, do not maintain an open error log, and lack a Human Verification Layer for automatic changes.

Transparency, Auditability, and Verification of Changes. Doshi-Velez & Kim [19], Ribeiro et al. [20], and Hind [21] emphasize that adaptive systems should maintain a change log and allow verification by a human operator. Shokri et al. [22] and Papernot et al. [23] demonstrate that without proper oversight and an audit trail, self-learning models can pose privacy and security risks.

An audit trail is considered a critical component for ensuring transparency, safety, and compliance with ethical and regulatory requirements, including the EU AI Act and OECD principles.

Additionally, [24] highlights the importance of adhering to secure coding principles during the design and implementation of software and information systems. Since AI is essentially an intelligent, software-implemented service, compliance with secure coding standards is not only advisable but also mandated by officially approved development requirements and guidelines.

Recent reviews underline the key risks associated with uncontrolled self-learning: accumulation of errors, deviation from user goals, unpredictability of autonomous strategies, loss of transparency, and ethical consequences. In this context, human verification acts as an arbiter and moderator, capable of assessing and halting incorrect self-learning trajectories.

Table 2
Gaps in Current Approaches

Human-Centered AI (HCAI)	Human-in-the-loop principles	Not integrated with adaptive NLP correction
Human Factor	Slips/lapses/mistakes models	Not used for adaptation rules
NLP Correction	Keyboard Layout Error (KLE), seq2seq models	Lacks audit trail and human verification
Continual Learning	Methodologies for lifelong learning	Safe pre-processing pipeline not applied
Industry Practices	FST decoders, transliteration	No error log or change control
Safety and Privacy	Theoretical protection models	Not integrated into self-learning systems
Human-Centered AI (HCAI)	Human-in-the-loop principles	Not integrated with adaptive NLP correction

Thus, the literature analysis shows that although numerous studies address human errors [1, 2, 7, 8], input correction algorithms [11-14], and human-in-the-loop approaches [3, 6], these efforts remain fragmented and are not integrated into the self-learning processes of large language models (LLMs). Industrial solutions demonstrate effectiveness in correcting isolated errors; however, they do not generate new rules based on large-scale repetitions and lack a transparent change log with human verification [19-21]. Similarly, continual learning methodologies [15-17] do not ensure safety for large models due to the risk of catastrophic forgetting.

This highlights a clear research gap: the development of an adaptive pre-processing mechanism that accumulates human errors in a system log, defines threshold conditions for generating new rules, proposes changes, routes them for human verification, and integrates the confirmed rules into the pre-processing pipeline. Implementing such an approach would combine the benefits of AI self-learning with oversight and transparency, reduce the risks of incorrect adaptation, and ensure compliance with the principles of human-centered artificial intelligence.

3. Research methods

Statistical analysis was conducted by testing AI services in conversational mode. The responses of AI models were examined when users entered typical human-factor errors, such as incorrect keyboard layouts, gibberish input, or omissions. The models' reactions during the first interaction were recorded and then compared with their behavior when similar errors were repeated within the same session and across subsequent sessions.

This approach made it possible to evaluate the consistency and stability of model behavior, detect any signs of adaptation, and analyze whether large language models retain information about user-specific errors or correction preferences over time.

4. Main material

4.1. Domain Analysis

Our examination of ChatGPT (OpenAI), Google Gemini, Claude (Anthropic), and Bing Chat / Microsoft Copilot revealed that large language models (LLMs) do not automatically correct keyboard-layout errors and fail to retain information about individual user errors or correction priorities across sessions.

During the initial interaction, most models did not even attempt to infer the cause of the error (only Google Gemini suggested a possible reason but did not provide translation suggestions and required the user to formulate the correct phrase themselves). When similar errors were repeated within a session, if the user had identified the cause and corrected it manually the first time, the models either did not adapt or were slow to respond to self-correction, typically only asking whether this was the same error as before. Only Google Gemini, a product from a company with experience implementing such corrections in search, proposed a correction automatically. However, this correct response did not persist in subsequent sessions, similarly to all other tested models. Claude (Anthropic) also realized, when tested in different conversational contexts, that the keyboard layout could be mixed up and tried to recognize it, but was unable to translate and recognize the phrase correctly.

Based on the explanations from the services themselves, LLMs do not automatically correct such errors for several reasons:

- Focus on content rather than input: The primary task of the system is to understand and respond to the query. If a user types "gjnhs,yj" by mistake, the system does not change it automatically because it could be a special code, abbreviation, or other meaningful input.
- Priority on accuracy: Automatic correction without confirmation can lead to misinterpretation. For example, if a user intentionally types a word in Latin script and the system "corrects" it to Cyrillic, this constitutes an error.
- Transparency and user control: The decision always remains with the user. The user explicitly indicates that it was a keyboard-layout error, and only then does the system adjust its response. This ensures that the system does not modify text without user consent.
- Diverse contexts: In search engines or autocorrect systems, the context is simpler — it is almost always obvious that "iphon" -> "iPhone." In LLM applications, contexts are highly varied, and automatic correction may distort the meaning of the query.

4.2. Conceptual Model of the Self-Learning Methodology: System Architecture.

4.2.1. Adaptive approach to human factor errors.

The proposed approach is proactive and grounded in the psychology of human behavior. Humans tend to make recurring errors, yet their cognitive system adapts: it remembers previous actions, takes context into account, and logically infers that a meaningless string may result from a keyboard layout shift (e.g., "ghbdtn" → "привіт", "studenka" → "студентка").

The proposed concept of the "human factor in AI" transfers this logic into an algorithmic form:

- Machine logic: "Did the user make a mistake? I will ask again."
- Human logic: "The user is repeating the same error. They likely forgot to switch the keyboard layout. I will apply a correction and keep it unless there is an objection."

This concept involves using Human Factor Global Log for such errors when they occur frequently.

If a Human Factor Global Log is introduced, which records recurring errors and includes their type, number of users, activation threshold, correction status, and update date, the following error-processing logic can be implemented:

1. In the first occurrence, the system asks for clarification.
2. The error is recorded in the error log.
3. In subsequent sessions, previous experience is taken into account.
4. When a critical number of similar cases is accumulated, the system proposes a new rule for human approval.

This mechanism ensures both adaptivity and human control over system evolution.

Differentiation of Error Types. It is essential to distinguish between different categories of errors:

Keyboard layout errors ("gibberish" input):

The system checks whether the entered symbols correspond to another keyboard layout and proposes a correction for confirmation. A global log is not required, as this is a technical error with deterministic correction rules. Example: "ghbdtn" → "привіт"

Spelling and grammatical errors:

The system accumulates examples and forms correction rules (e.g., "птреба" → "потрібно"). Each proposed rule must undergo user verification. A global log is necessary to enable generalization and controlled adaptation.

Table 3

Comparison of Approaches

Error Type	Processing Logic	Global Log Required
Keyboard layout errors	Translation + user verification	Not required
Spelling / grammar errors	Accumulation -> rule generation -> approval	Required

Therefore, for technical errors caused by keyboard layout, a simple "transliteration -> verification" mechanism is sufficient and does not require recording entries in a system-level error log. By contrast, spelling and grammatical errors demand a systemic approach: they should be recorded, aggregated, and formalized into rules that undergo human verification. This combined strategy allows the system to remain flexible — avoiding overload with irrelevant data while accumulating useful knowledge for more complex cases — and thereby ensures controlled and transparent self-learning.

4.2.2. Justification of Risks and the Necessity of Human Verification. Human Verification Layer.

Previous literature analysis has shown that uncontrolled self-learning in AI carries several risks - ranging from error accumulation and deviation from intended goals to loss of transparency and ethical violations. All these issues share a common denominator: the absence of systematic human verification. Therefore, the next step is to design a mechanism that integrates human verification into the process of creating new rules.

In practical terms, this means that any automatically generated AI suggestions for corrections or new rules cannot be integrated without prior human review. This approach helps prevent two critical threats:

1. False self-learning: when the algorithm misinterprets slang or local peculiarities as a "widespread error."
2. Adaptation to malicious actions, such as bot attacks or deliberate (template) entry of meaningless data to create the illusion of a widespread error.

Just as a doctor verifies a diagnosis before prescribing treatment, an AI developer must confirm or reject automatic algorithmic corrections. This ensures a balance between system autonomy and safe operation.

It is proposed to solve this task with the Human verification Layer(HVL). We propose

introducing it as a key component of the safe self-learning architecture. This module acts as a "filter," verifying automatically generated suggestions before they are integrated into the system.

The protection logic for the global error log relies on:

Two-Stage Verification

- Stage 1: Users confirm or reject the suggestion.
- Stage 2: The developer or moderator reviews new rules before they become global.

Levels of Trust. Each rule is assigned a trust rating:

- Low: a new rule confirmed by only a few users.
- Medium: confirmed by many users but not yet verified by a developer.
- High: confirmed by both users and developers -> becomes global.
- Protection Against Attacks
- Anomalies: If a new rule is confirmed too quickly or massively from a single IP/bot, the system flags it as "suspicious."
- Logging: All confirmations are stored anonymously but include attributes to trace the source of potential attacks, following principles similar to those established for guaranteed information systems [33].

Moderation: Developers have an interface to review, adjust, and verify "proposed" rules.

Here, for example, for keyboard layout errors, we can immediately offer the following algorithm: the system suggests a probable word for confirmation if such a word exists in the user's native language, which they are typing in, and we record the number of confirmations in response to such a suggestion, rather than the words themselves. This addition makes the system resistant to attacks: users help to quickly correct errors and establish the correctness of this strategy for correcting such types of errors as a new rule, but the final decision on the global rule is made by a human developer. This creates a balance between the speed of self-learning and security.

To implement the concept, it is necessary to follow the logical scheme shown in Table 4.

Table 4

Global Log Logic

Stage	Action	Performed By	Outcome
1. Accumulation	The system records a "garbled input" or a spelling/grammar error	Automated	An entry is created in the global log
2. Proposal	AI suggests a correction or translation	Automated	The user sees the suggested variant
3. User Verification	The suggestion is confirmed or rejected	User	A "candidate rule" is formed
4. Developer Review	New rules are reviewed; suspicious entries filtered	Human developer	Verified rule becomes global
5. Global Log	Storage of verified rules	System + developer	Rules are available for all users
6. Update	Removal or correction of suspicious rules	Developer	Protection against attacks and incorrect adaptations

Adaptive System Logic. When a user inputs text, the system follows these steps:

1. The system verifies the user's error profile (if the user has allowed it to be maintained).
2. If the error is already recorded in the user's personal profile and the correction has been previously confirmed, the system automatically applies the correction.
3. If the error exists in the global Human Factor Log, the system attempts to correct it automatically and presents the suggestion to the user for confirmation.
4. If the error is new, the system queries the user about it and adds it to the Log.
5. Global Log stores errors and statistics for the implementation of subsequent steps
6. in generating new rules for their confirmation and verification.

4.2.3. From Statistics to Knowledge.

The algorithm is based on a knowledge base development methodology: accumulation of error statistics -> transformation into generalized "knowledge." Every rule must still be verified by a human.

Statistics:

- By default, the system collects anonymous statistics: frequencies, averages, minimums, and maximums.
- The Human Factor Log records only the type of error and the applied correction method.
- Users may enable a personal profile mode, where their individual choices and confirmations are stored for personalized adaptation.
- Knowledge:
- Knowledge represents generalized rules or patterns derived from the collected statistics and can be used for decision-making.
- Logic of transition: statistics -> analysis -> knowledge.
- Once an activation threshold is reached, the system formulates a generalized rule and submits it for human verification.

Change Log for Algorithmic Adaptations. When new rules are added, a Change Log is maintained, with each entry containing:

- Rule identifier
- Date and time of the proposal
- Number of users who triggered the error
- Content of the proposed transformation
- Verifier decision (accepted / rejected / modified)
- Reason for the decision
- Algorithm version after the change

The "reason for decision" field ensures transparency, auditing, and version rollback capability. It allows tracking why a rule was accepted or rejected (e.g., due to a bot attack or conformity with orthographic norms). This documentation aligns with ISO/IEC AI Governance standards, which mandate justification of changes in self-learning systems.

5. Architecture of a Self-Learning System for Correcting Human-Factor Errors

Based on the previous analysis, we propose an architecture for a self-learning system designed to detect, accumulate, and correct errors caused by human factors, particularly text input mistakes due to incorrect keyboard layouts. The system operates by collecting statistics, incorporating user verification, and integrating changes in a controlled manner into the Input Preprocessor module-without modifying the core of the large language model (LLM).

The system functions as a multi-level knowledge base that:

- Quickly responds to user errors
- Accumulates data for self-learning
- Provides protection against manipulations through human verification

All automatically generated rules are first stored in a candidate update log. An expert human reviews and approves each modification before integration.

This mechanism ensures:

- Transparency of algorithmic changes
- Protection against mass attacks
- Auditability and reproducibility of system evolution
- Compliance with human-centered AI principles

Each entry in the change log includes:

- Timestamp
- Error pattern
- Statistical justification
- Proposed transformation
- Human decision (accepted / rejected / modified)

Component Structure of the Self-Learning System, described in Table 5.

Table 5
Main Components of the Self-Learning System

Component	Purpose	Function
Persistent Memory	Long-term storage of detected errors and user confirmations across sessions	Enables recognition of repeated errors and formation of a knowledge base
Human Error Log (HEL)	Centralized log of human-factor cases with metadata and user reactions	Provides analytics of all users' errors and statistical insights
Input Correction Layer (ICL)	Intercepts input, suggests corrections, records user response	Ensures transparency, modularity, and accumulation of only verified cases
Adaptive Processing	Triggered when a threshold for a specific error type is exceeded	Automatically incorporates the verified rule into standard processing logic

Logical Model of the Self-Learning System. The functional model of self-learning management is presented in Table 6.

Table 6
Functional Purpose of Modules in the Self-Learning System

№	Module	Function
1	Input Analyzer	Analyzes user input to detect potential errors
2	Keyboard/Layout Detector	Identifies errors caused by incorrect keyboard layout
3	Human Error Log (HEL)	Records occurrences, stores metadata, and logs user confirmations
4	Threshold Engine	Calculates trust score and determines whether a new rule should be generated and verified before deployment
5	Human Verification Layer (HVL)	Provides manual verification of new rules by a human
6	Model Adapter	Integrates verified rules into the input pre-processing module

System Workflow

1. Input Analyzer
   - Performs initial syntactic and statistical analysis of user input.
   - Detects patterns of errors typical for incorrect keyboard layouts or random text input.
2. Keyboard/Layout Detector
   - Compares characters across different keyboard layouts (e. g., Latin → Cyrillic).
   - Determines the most likely transformation of the input text.
3. Human Error Log (HEL)
   - Stores a standard set of attributes for each detected error:
     - Timestamp (date and time)
     - Error type
     - Anonymized user identifier
     - Unique contextual identifier
     - Number of repetitions
     - Whether the user corrected the input
   - Provides the basis for statistical analysis in the Threshold Engine module.

4. Interface Correction Layer (ICL). The Interface Correction Layer (ICL) is a dedicated architectural module that intercepts user input and performs the following functions:
   - identifies error patterns (e.g., "gibberish" text);
   - proposes candidate corrections to the user;
   - records the user's response (confirmed / rejected);
   - forwards only the agreed and validated text to the system core.

The main advantages of the ICL include:
   - modularity and transparency of the correction process;
   - scalability to new types of corrections (e.g., auto-replacements, abbreviations, emojis);
   - safety, ensured by the centralized accumulation of only user-confirmed cases.

5. Threshold Engine. The Threshold Engine is activated when the following conditions are met:

- a sufficient number of homogeneous errors has been accumulated (the threshold is computed using a threshold function that accounts for the error type, e.g., 1000 occurrences);
- there is a sufficient number of user-confirmed corrections or responses indicating the correctness of the proposed transformation.

As a result, a candidate rule is generated and forwarded for further human verification.

6. Human Verification Layer (HVL). The Human Verification Layer (HVL) provides human-centered control over the self-learning process. Its core functions include:

- receiving and registering automatically generated rules;
- enabling developers to review, comment on, and edit proposed rules;
- supporting decision-making: approve, reject, or modify a rule;
- maintaining an audit of previous modifications and enabling rollback.

The HVL log stores:

- timestamps;
- a description of the error;
- statistical justification;
- the proposed transformation;
- the human decision.

This mechanism prevents uncontrolled changes, protects the system from attacks based on massive injection of incorrect inputs, and ensures the reproducibility and traceability of rule evolution.

7. Model Adapter. The Model Adapter integrates verified rules into the pre-processing pipeline without modifying the core LLM. This design allows the system to:

- preserve the generality and stability of the base model;
- update correction algorithms in a modular manner;
- avoid costly retraining procedures.

As a result, the main AI Core (LLM) receives only sanitized, consistent, and user-validated input text. The overall operational scheme of the self-learning system for correcting errors caused by the human factor is shown in Figure 1.

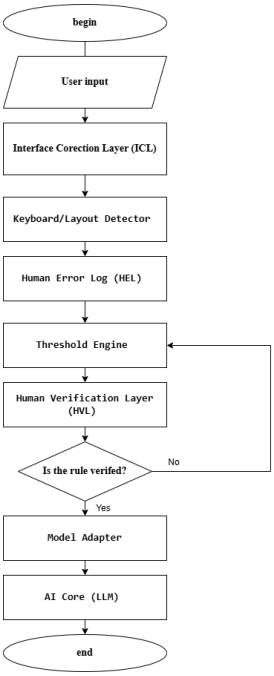

Figure 1: Architecture of the Self-Learning System

It is important to emphasize that both the inclusion of a dedicated human verification rule-creation module and the placement of the entire system at the interface layer ensure compliance with secure coding requirements and established software development standards [24, 25]. Moreover, research on automated testing systems for identifying vulnerabilities in large language models [26] further highlights the importance of rigorous verification mechanisms when deploying adaptive AI systems. This architectural decision isolates adaptive behavior from the LLM core and

guarantees that all modifications are mediated through controlled, auditable, and formally validated processes.

To implement the proposed methodology, we introduce an approach for computing verification thresholds based on a rule activation function. This function depends on the type of error, the number of observed instances, and the number of user confirmations. In addition, it incorporates different statistical regimes:

- For a small number of observations, a Bayesian estimation framework is applied to account for uncertainty and to avoid premature generalization from sparse data.
- For large concentrations of cases and rapid growth in confirmation rates, anomaly and attack detection mechanisms are activated to identify potentially malicious or automated behavior.

These considerations are aligned with modern methods of ensuring information protection in cybersecurity systems [32]. Bayesian inference thus serves as the mathematical core of the threshold mechanism, enabling the system to decide whether a candidate rule is sufficiently reliable to be forwarded for human verification. This approach ensures a principled balance between responsiveness and caution: rules are neither generated too early based on insufficient evidence nor accepted blindly under conditions that may indicate coordinated attacks or data poisoning attempts.

6. Conclusions

This work implements the principles of Human-Centered Artificial Intelligence (HCAI), focusing on supporting user self-efficacy and accounting for their real behavior during system interaction. The proposed model combines adaptive self-learning mechanisms with mandatory human oversight, meeting modern requirements for safe and reliable intelligent information systems.

The developed system can be viewed as the foundation of an intelligent information platform functioning as a multi-layer knowledge base, providing:

- rapid response to user errors;
- accumulation and generalization of data for subsequent self-learning;
- protection against manipulation and incorrect modifications through human verification of rules.

This approach advances beyond most existing solutions, which typically focus only on adaptation to cognitive models of users and do not consider individual behavioral patterns, such as typing errors, keyboard layout issues, or user-specific interaction patterns.

The proposed concept implements the principle of converting error statistics into rules, analogous to knowledge-based systems, while considering the frequency of user confirmations. Bayesian estimation is used as the core of the threshold mechanism, allowing informed decisions on generating new rules and determining whether they should be submitted for human verification. This ensures system robustness under conditions of both sparse observations and high concentrations of events or potential anomalies.

Protection against incorrect or malicious self-learning is achieved through:

- verification of new rules by a human;
- isolating behavior correction mechanisms in the user interaction module rather than in the core AI management modules, significantly reducing the risk of critical errors or undesired changes in system logic.

The results provide a methodological and architectural foundation for the further development of intelligent information systems capable of safe, adaptive, and transparent self-learning while accounting for the real logic, behavior, and errors of users. The principles underlying this work are consistent with advances in personalized model adaptation for domain-specific tasks [30] and the use of digital forensics techniques leveraging multimodal large language models [31], fully aligning with the modern paradigm of human-centered artificial intelligence.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] J. Reason, Human Error, Cambridge University Press, 1990.

[2] S. Dekker, The Field Guide to Understanding Human Error, Ashgate, 2017.

[3] S. Amershi, et al., Guidelines for Human-AI Interaction, in: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, ACM, New York, NY, 2019. doi:10.1145/3290605.3300233.

[4] Y. Li, M. Hoyle, S. Robinson, D. Budd, P. Hui, Human factors in AI safety: A survey, IEEE Trans. HumanMachine Syst., 52 (2022) 1006-1019.

[5] E. Glikson, A. W. Woolley, Human trust in artificial intelligence: Review of empirical research, Academy of Management Annals 14, 2020, 627-660. doi:10.5465/annals.2018.0057.

[6] Y. Li, X. Gu, A Risk Framework for Human-centered AI in Education, Comput. & Educ., 193 (2023) 104674.

[7] J. Reason, The Human Contribution: Unsafe Acts, Accidents and Heroic Recoveries, Cambridge University Press, 2008.

[8] E. Hollnagel, Safety-I and Safety-II: The Past and Future of Safety Management, Ashgate, 2014.

[9] D. Norman, The Design of Everyday Things, MIT Press, 2013.

[10] B. Shneiderman, et al., Designing the User Interface: Strategies for Effective Human-Computer Interaction, 6th ed., Pearson, 2016.

[11] R. Caruana, Adaptive character-level models for noisy text input, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Association for Computational Linguistics, 2015, pp. 2146-2151. doi:10.18653/v1/D15-1170.

[12] Y. Belinkov, Y. Bisk, Synthetic and Natural Noise Both Break Neural Machine Translation, in: Proceedings of the 6th International Conference on Learning Representations, ICLR, 2018. doi:10.48550/arXiv.1804.07755.

[13] A. Golding, D. Roth, A winnow-based approach to context-sensitive spelling correction, Mach. Learn., 34 (1999) 107-130. doi:10.1023/A:1007666700385.

[14] V. Balyan, S. Jangid, A. Jain, K. S. Raju, Keyboard Layout Correction using Language Models, IEEE Access 9 (2021) 20012-20022. doi:10.1109/ACCESS.2021.3052731.

[15] G. Parisi, R. Kemker, J. Part, C. Kanan, S. Wermter, Continual Lifelong Learning with Neural Networks: A Review, Neural Networks 113 (2019) 54-71. doi:10.1016/j.neunet.2019.01.012.

[16] M. Delange, et al., A Continual Learning Survey: Defying Forgetting in Classification Tasks, IEEE Trans. Pattern Anal. Mach. Intell. 44 (2021) 3366-3385. doi:10.1109/TPAMI.2021.3074692.

[17] Z. Chen, B. Liu, Lifelong Machine Learning, 2nd ed., Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2018. doi:10.2200/S00839ED1V01Y201804AIM037.

[18] W. Zhang, Y. Ning, N. Yu, Y. Chen, L. Ma, Development and validation of the artificial intelligence self-regulated learning scale (AI-SRLS) for teachers, Educ. Inf. Technol., 30 (2025) 22989-23013. doi:10.1007/s10639-025-13670-x.

[19] F. Doshi-Velez, B. Kim, Towards a Rigorous Science of Interpretable Machine Learning, 2017.

[20] M. T. Ribeiro, S. Singh, C. Guestrin, Why Should I Trust You? Explaining the Predictions of Any Classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016, ACM, New York, NY, 2016, pp. 1135-1144. doi:10.1145/2939672.2939778.

[21] M. Hind, Explaining Explainable AI, XRDS: Crossroads, The ACM Magazine for Students 25 (2019) 16-19. doi:10.1145/3363674.

[22] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership Inference Attacks Against Machine Learning Models, in: Proceedings of the 2017 IEEE Symposium on Security and Privacy, SP 2017, IEEE Computer Society, Washington, DC, 2017, pp. 3-18. doi:10.1109/SP.2017.41.

[23] N. Papernot, P. McDaniel, A. Sinha, M. Wellman, SoK: Security and Privacy in Machine Learning, in: Proceedings of the 2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, IEEE, 2018, pp. 399-414. doi:10.1109/EuroSP.2018.00035.

[24] Y. Momryk, Y. Yashchuk, R. Tuchapskyi, Professional approach as a method of information protection at the stages of relational database development and software for working with them, Cybersecurity: Educ. Sci. Technol. 3 (23) (2024) 42-55. doi:10.28925/2663-4023.2024.23.4255.

[25] Y. Momryk, D. Sabodashko, Numeric Fields in Database Development: From Optimal Design to Secure Coding — SQL Attacks Protection Method, CEUR Workshop Proc., 2025, 84-96.

[26] V. Khoma, et al., Investigation of vulnerabilities in large language models using an automated testing system, CEUR Workshop Proc., 3826 (2024) 220-228.

[27] O. Deineka, et al., Detection confidential information by large language models, Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska 15 (3) (2025) 91-99. doi:10.35784/iapgos.6910.

[28] O. Deineka, O. Harasymchuk, A. Partyka, A. Obshta, Application of LLM for Assessing the Effectiveness and Potential Risks of the Information Classification System According to SOC 2 Type II, CEUR Workshop Proc., 3991 (2025) 215-232.

[29] A. Zhuravchak, A. Piskozub, B. Skorynovych, Y. Lakh, D. Zhuravchak, P. Hlushchenko, P. Venherskyi, I. Beliaiev, M. Vorokhob, I. Kolbasynskyi, Design and development of a large language model-based tool for vulnerability detection, EasternEuropean J. Enterp. Technol. 2, (2(134)) (2025) 75-83. doi:10.15587/1729-4061.2025.325251.

[30] V. Brydinskyi, et al., Enhancing Automatic Speech Recognition With Personalized Models: Improving Accuracy Through Individualized Fine-Tuning, IEEE Access 12 (2024) 116649-116656. doi:10.1109/ACCESS.2024.3443811.

[31] T. Fedynyshyn, et al., Leveraging Multimodal Large Language Models for Digital Forensics in Military Personnel Detection in Mobile Device Images, in: Proceedings of SNPD 2025-Summer, IEEE, 2025, pp. 127-132. doi:10.1109/SNPD65828.2025.11253944.

[32] O. Harasymchuk et al., Modern methods of ensuring information protection in cybersecurity systems using artificial intelligence and blockchain technology: collective monograph, Technology Center PC, Kharkiv, 2025. doi:10.15587/978-617-8360-12-2.

[33] H. Hulak, L. Kriuchkova, P. Skladannyi, I. Opirskyy, Formation of requirements for the electronic record-book in guaranteed information systems of distance learning, CEUR Workshop Proc., 2021, 137-142.