

Supporting Rule-Based Control with a Natural Language Model [†]

Martin Kernács ¹ and Olivér Hornyák ^{2,*} ¹ Hatvany József Doctoral School, University of Miskolc, Egyetem u. 1, 3515 Miskolc-Egyetemváros, Hungary; martin.kernacs@uni-miskolc.hu² Institute of Information Science, University of Miskolc, Egyetem u. 1, 3515 Miskolc-Egyetemváros, Hungary

* Correspondence: oliver.hornyak@uni-miskolc.hu

[†] Presented at the Sustainable Mobility and Transportation Symposium 2025, Győr, Hungary, 16–18 October 2025.

Abstract

The usage of Artificial Intelligence (AI) in control loops and rule-based frameworks is a novel approach in automation and decision-making processes. Large Language Models (LLMs) are redefining conventional rule-based systems by introducing intuitive natural language interfaces, drastically changing the creation of rules, and minimizing operational complexity. Unlike static controllers, AI-enhanced systems can autonomously evolve with real-time environmental changes, achieving optimal performance without manual intervention. By allowing non-experts to modify rules through natural language commands, LLM can change the control system management. These advancements not only improve adaptability and operational efficiency but also reduce downtime through proactive error detection and self-correction mechanisms. AI-powered systems allow refining operations, thus accelerating response speeds and increasing reliability. The synergy between rule-based logic and AI-driven intelligence provides a new approach for autonomous systems, improving their capability of context-specific decision-making. In this paper, an approach is presented to control a storage system by natural language commands. The comparison of the Hungarian and English language interpretations is discussed.

Keywords: natural language model; large language models; controls



Academic Editors: Szabolcs Kocsis-Szürke, Dániel Csikor and Boglárka Eisinger Balassa

Published: 10 November 2025

Citation: Kernács, M.; Hornyák, O. Supporting Rule-Based Control with a Natural Language Model. *Eng. Proc.* **2025**, *113*, 56. <https://doi.org/10.3390/engproc2025113056>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Rule-based control systems play an important role in industrial automation and business process management [1]. However, defining and maintaining rules is often a time-consuming and expertise-demanding task. Recent advancements in Natural Language Processing (NLP) models [2], particularly Large Language Models [3] such as ChatGPT [4] and Llama [5], offer new opportunities to simplify rule definition and management through natural language interfaces.

In this paper, we present an approach in which a natural language model supports the operation of a rule-based controller. The system allows users to define or modify rules in natural language, which are then translated into formal rules by an automatic interpreter component. This approach aims to reduce development time and make rule-based control more accessible to non-technical users. Our motivation is to create a model that is

equipped with prompt-based control [6], and the input tasks are given in NLP. Furthermore, we introduce an evaluation algorithm that assesses system behavior based on the generated responses, ensuring reliability and correctness.

Related Work

The integration of Large Language Models (LLMs) into control systems represents a growing area of interest, particularly in the context of bridging symbolic logic with natural language interfaces. Early studies in rule-based AI control systems, such as Taylor and Stringer [7], explored expert systems for synthesizing nonlinear control logic, laying the groundwork for automated reasoning frameworks. With the advent of transformer-based models, LLMs like GPT and LLaMA have demonstrated strong capabilities in parsing and generating structured outputs from natural language, enabling applications in code generation, configuration, and rule synthesis [8].

A key advance in LLM reasoning has been the introduction of Chain-of-Thought (CoT) [9] prompting, which guides models through intermediate steps to improve performance on multi-step reasoning tasks [10]. CoT has been successfully applied in tasks requiring arithmetic logic, symbolic processing, and question answering, as shown by Wei et al. [10] and further refined in variants like Tree-of-Thought and Self-Consistency prompting. However, most CoT implementations focus on abstract reasoning rather than deterministic rule translation required for industrial control systems.

Recent work by Servantez et al. [10] introduced the concept of Chain of Logic (CoL), an LLM-compatible framework for rule-based reasoning with formal logic structures. Their approach demonstrates how structured prompts [11,12] can guide LLMs to perform accurate logical transformations, providing a more deterministic behavior suited to applications like planning and decision trees. While most existing studies emphasize the reasoning capabilities of LLMs, few explore their use in direct control over physical systems, particularly in Programmable Logic Controller (PLC) environments. Xia et al. [13] presented a framework for using LLMs to control industrial automation systems, yet their focus remains largely on system integration rather than fine-grained rule mapping or failure analysis.

2. Proposed Approach

To assess the effectiveness of our proposed NLP-supported rule-based control system, we conducted a time-based comparative analysis against a conventional PLC-based control setup [14,15]. Our research hypothesis is that incorporating natural language support can significantly reduce the time required to create and modify control rules, particularly for non-expert users, while maintaining the operational accuracy and efficiency of the control system.

In traditional PLC systems, modifying control logic—even for minor changes—may require recompiling the program and reloading the executable onto the controller. Depending on the extent of the changes, this process may also result in system downtime. In contrast, our architecture separates control logic from the PLC program by externalizing rule definition. The PLC handles peripheral device communication only, while all logic-related tasks are delegated to a rule-based controller. This design allows rule modifications to be performed in natural language [16]. The only adjustment required is to the input prompt provided to the controller—eliminating the need for PLC code modification. This feature is particularly useful in industrial environments, where on-demand changes to equipment behavior are frequently requested by operators, even when the system is functioning reliably. Only communication needs to be written within the PLC program, which covers all peripherals. When defining rules in natural language, only the prompt of the currently running program needs to be adjusted according to requirements. In a real industrial environment, such modifications frequently occur even in well-functioning and reliable equipment at the operator's request.

To evaluate the language-agnostic capabilities of the system, we tested it using prompts written in both English and Hungarian. The responses generated by the controller were

analyzed for semantic correctness and functional reliability. Furthermore, to enhance output consistency and reasoning accuracy, we incorporated a step-wise validation approach using a technique known as Chain of Logic. This method enforces structured reasoning in rule interpretation, helping ensure that the system consistently produces valid and predictable control logic.

The experimental setup simulates an industrial control environment using a combination of hardware and software tools. Programming was performed in Visual Studio Code Version:1.105.1 using Python v3.10, which manages communication with a Siemens S7-1500 PLC over the Modbus TCP protocol. The system uses the pre-trained LLaMA 3.1 model to interpret user instructions and convert them into control actions. Communication with the PLC is handled programmatically, and structured data exchange is achieved using JSON formatting. A simulated environment is provided by Factory IO, which models conveyor systems, actuators, and sensors in 3D, allowing safe validation of control logic before real-world deployment. This modular architecture provides flexibility and supports future expansion of NLP-enhanced control capabilities in industrial settings.

3. Model of Rule-Based Control with NLP

This paper presents a control model for a system that includes roller tracks and route switches. The goal is to enable users to control the system using natural language commands, reducing the need for traditional programming interfaces. The system is designed to direct packages to one of seven possible storage locations. By specifying a destination in natural language (e.g., “Storage location 1”), the control system activates the appropriate rollers and switches to guide the package along the correct route. For example, if the user specifies “Storage location 1,” the system interprets this input, calculates the required path, and actuates the necessary components. In this case, the package would start from the initial point, turn left at the first switch, and continue straight to the final destination. Users can also specify operational parameters like speed, e.g., “Storage location 1, 50% speed” instructs the rollers to operate at half speed. Figure 1 illustrates the model layout. Arrows indicate possible routes, and “WSW” labels represent way switches.

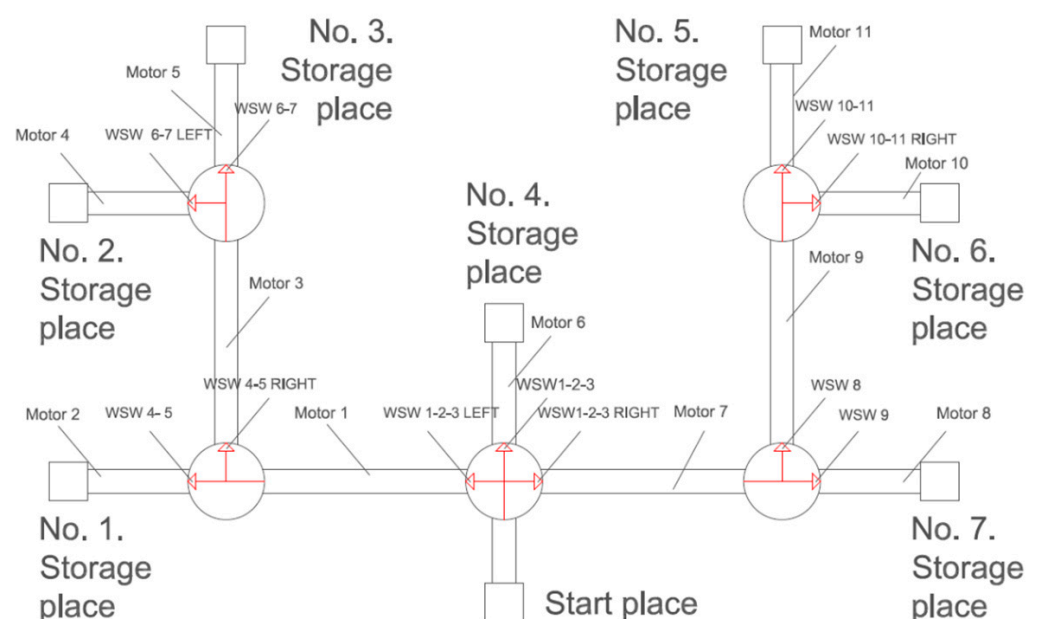


Figure 1. Model outline.

A simulation of the system is implemented using a software called Modbus Slave. The operational instructions of the tasks required for Modbus communication are defined in

an external file (tasks.txt), which maps natural language commands to Modbus register values. This file-based interface makes it possible to update system behavior without directly modifying the underlying PLC or Python code.

Our goal is to reduce machine downtime and process time. We aim to achieve this by allowing new requests to be implemented in the form of an easily editable file, hence the use of the file. This file will contain only natural language (NLP-based) instructions [17], eliminating the need for complex programming knowledge to operate the system. The solution enables users to modify operational parameters effortlessly without directly editing the control software. As a result, we provide a more flexible and efficient system operation.

As illustrated in Figure 1, each storage location is associated with a specific route composed of a sequence of motors and way switches. To direct a package to Location 1, the system activates the following components: WSW 1-2-3, WSW 1-2-3 LEFT, Motor 1, WSW 5-6, and Motor 2. For more complex control tasks, the system supports full access to all relevant control variables, including motor states and way switch positions. Within the rule-based control framework, each of these variables is mapped to specific Modbus registers, which the controller can write to or read from to update the system state. Switch controllers are represented using a 16-bit register where each bit corresponds to a specific control output (e.g., a switch or motor). This bit-level mapping allows efficient use of register space. These outputs correspond to specific motors and way switches involved in routing the package along the required path. The controller calculates the required bit pattern, encodes it into the appropriate Modbus register value, and sends the command to the PLC.

This setup allows the controller to use precalculated register values, simplifying execution. However, it introduces a limitation: if outputs need to be changed—such as adding a new actuator like an indicator lamp or buzzer—the register value must be recalculated manually, unless a dynamic method is used.

To overcome this, our system leverages the Large Language Model (LLM) to recompute register values automatically when new outputs are defined. Users can simply update the task mappings file with the new output description, and the LLM calculates the appropriate binary register value that activates the desired outputs. This method eliminates the need to manually modify the PLC program, as long as the additional outputs are already addressable via available bits. Essentially, each new output is assigned to an unused bit, preserving flexibility while avoiding low-level reprogramming.

The proposed workflow is as follows:

1. A user provides a natural language prompt as the system's input for the system.
2. The prompt is processed by the LLaMA 3.1 large language model.
3. The model interprets the user's instruction.
4. A predefined list of steps written in JSON format is loaded. Each entry in the file maps a natural language action to its corresponding register value and control logic.
5. The model finds the appropriate predefined action for the user's request.
6. Based on the matched action, the system retrieves the associated numerical value required for execution.
7. This value is written into a specific Modbus register that is monitored by the PLC.
8. The PLC reads the value and performs the corresponding control operation.

This process enables natural language-based control of PLC systems via Modbus communication. Figure 2 shows the information flow of the system.

This process enables natural language-based control of PLC systems via Modbus communication.

The rules file contains a JSON (JavaScript Object Notation) [18]-formatted regulations.

This is an appropriate format for the LLM model, so that it can compute the register values based on the desired output states.

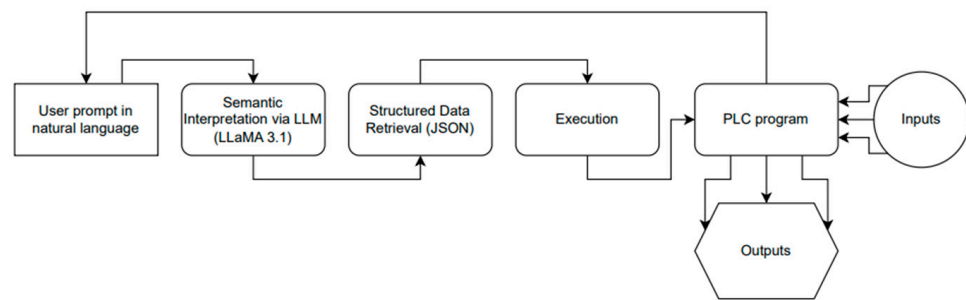


Figure 2. Model flowchart.

4. Model Environment

The development and testing environment was designed to integrate natural language processing with industrial control systems. All programming was performed in Visual Studio Code, using Python 3.10 as the primary language. The natural language interface was powered by LLaMA 3 large language model (in its pre-trained form), which interprets user instructions and maps them to control actions. For the physical control layer, a Siemens S7-1500 PLC (Siemens Zrt., Budapest, Hungary) was used. This controller is well-suited for industrial applications due to its robustness, modular architecture, and support for Modbus communication. System simulations were conducted in Factory IO (v2.5.8), which provided a realistic 3D environment for modeling conveyor systems, actuators, and sensors. Figure 3 depicts the model. This allowed for safe and controlled testing of the AI-driven control logic before deployment to the physical PLC. The integration of these components enabled the seamless execution of natural language commands in a simulated industrial environment, bridging AI with programmable logic control.

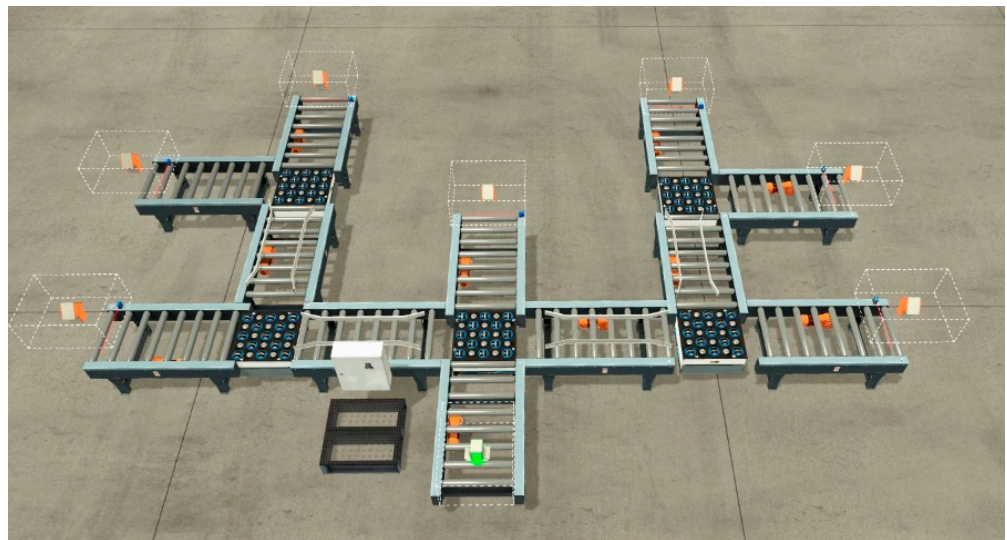


Figure 3. Model in Factory IO.

5. Expectations

In our work, we aim to develop an algorithm and a program that allows the model to be controlled exclusively through natural language [19]. The system relies on processing and interpreting user prompts, ensuring intuitive and efficient control. A well-written PLC program that properly manages the registers ensures that control requests written in

natural language function correctly. This means that the first and most crucial task is always writing the PLC program, and only afterward can the advantages of artificial intelligence and large language models (LLM) be utilized. In Table 1 we can see the answers for user prompts are not always correct. To use this model in any application, the model must be able to handle the user prompt correctly at any time. To address this challenge, we employ a reasoning strategy called Chain of Logic (CoL) to improve the stability and correctness of the model's outputs. The CoL process is dependent on the structure of the user prompt and the underlying logical rules of the control system. It consists of the following six steps:

1. Structured Input—Parse and preprocess the natural language prompt
2. Rule Decomposition—Break down the command into logical subcomponents
3. Logical Expression—Translate subcomponents into formal rules or expressions
4. Question Answering—Resolve the intended user request based on logic
5. Element Recomposition—Reconstruct the complete control instruction
6. Expression Resolution—Generate the final register value for Modbus execution

Table 1. Model answers for user prompts.

No.	User Prompt	Answer	Evaluation
1.	Take the box to 1 storage unit with 55% (voice)	Register 0<- 32779 Register 1<- 1 Register 11<- 5500	Registers are correct Values are correct Speed is incorrect
2.	Vidd a dobozt az 1-es tárolóhelyre, 55%-os sebességgel	Register 0<- 32779 Register 1<- 1 Register 80<- 5500	Registers are correct Values are correct Speed is incorrect
3.	Vidd az 1-es tárhelyre, 55%-al	Register 0<- 32779 Register 1<- 1 Register 12<- 137	Registers are correct Values are correct Speed is incorrect
4.	Take the box to second store at 89% speed (voice)	Register 0<- 33003 Register 1<- 6 Register 12<- 89	Registers are correct Values are correct Speed is correct
5.	Vidd a dobozt az 2-es tárolóhelyre, 89%-es sebességgel	Register 0<- 33003 Register 1<- 6 Register 12<- 89	Registers are correct Values are correct Speed is correct
6.	Vidd az 2-es tárhelyre, 89%-al	Register 0<- 33003 Register 1<- 6 Register 12<- 89	Registers are correct Values are correct Speed is correct
7.	Take the box to 3rd storage unit at 22% speed (voice)	Register 0<- 32875 Register 1<- 10 Register 12<- 80	Registers are incorrect Values are correct Speed is incorrect
8.	Vidd a dobozt az 3-as tárolóhelyre, 22%-os sebességgel	Register 0<- 32875 Register 1<- 10 Register 12<- 55	Registers are correct Values are correct Speed is incorrect
9.	Vidd az 3-as tárhelyre, 22%-al	Register 0<- 32875 Register 1<- 10 Register 12<- 17	Registers are correct Values are correct Speed is incorrect

Let us use CoL for the user prompts in our Table 1 and then check what answers we receive back.

No. 7:

Step 1: Is LLM understood and transformed correctly from the user prompt to the controller?

The prompt is correct if:

- the registers are correct
- the values are as given for the storage unit
- the speed is correct
- due the Modbus write there was no problem or collision

Step 2:

x_1 : corresponding register
 x_2 : appropriate values for the storage number
 x_3 : correct speed
 x_4 : Modbus can be written

Step 3:

$$\text{The control is reliable} = x_1 \wedge x_2 \wedge x_3 \wedge x_4. \quad (1)$$

Step 4:

x_1 : Q: Were the appropriate registers used?

A: No, depending on the task2.txt the corresponding registers are 0,1, and 13, but the answer was register 12. ✗

x_2 : Q: Are the values corresponding to the 2nd storage unit?

A: Yes, depending on the calculating logic rule: register 0 → 32875, register 1 → 10. ✓

x_3 : Q: Is the speed correct?

A: No, in the user prompt was 22%, but the answer was 80%. ✗

x_4 : Q: Was everything OK when writing the registry?

A: Yes, there was no error. ✓

Step 5:

The control is stable = ✗ $\wedge$ ✓ $\wedge$ ✗ $\wedge$ ✓

Step 6:

The model did not responded appropriately to the user request.

According to the CoL inspection, our model is not reliable. We use the Correctness Rate to measure and compare the answer with the user command. We can just accept when the rate is 1.

$$\text{Correctness Rate} = (\omega_1 y_1 + \omega_2 y_2) / (\omega_1 + \omega_2) \quad (2)$$

where

ω_1 : weighing for y_1 , in our case $y_1 = 0.5$,

ω_2 : weighing for y_2 , in our case $y_2 = 0.5$,

y_1 : storage unit corresponding, true = 1, false = 0,

y_2 : speed corresponding, true = 1, false = 0,

So, we make CoL algorithm for this LLM-based model.

We inspect the system for “Take the box to 3rd storage unit with 22% speed” user prompt.

Chain of Logic Inspection 1

Step 1: Checking LLM interpretation: Storage = 3, Speed = 80,

Step 2: Decomposed checks:

x_1 : True (LLM parsed valid numbers from prompt),

x_2 : True (Storage number matches prompt and exists in task list),

x_3 : False (Speed matches prompt and within 0–100),

x_4 : True (Modbus writable assumed),

Step 3: Logical expression: $x_1 \text{ AND } x_2 \text{ AND } x_3 \text{ AND } x_4 = \text{False}$,

Step 4: Answering sub-questions:

Q1: Did LLM extract numbers? A1: True,

Q2: Is storage number valid and matching? A2: True,

Q3: Is speed matching and in valid range? A3: False,

Q4: Can we write to Modbus? A4: True,

Step 5: Populating expression: True AND True AND False AND True,

Step 6: Final result: False,

$$\text{Correctness Rate} = (0.5 \cdot 1 + 0.5 \cdot 0) / (0.5 + 0.5) = 0.5 \quad (3)$$

In step 1, the numbers are valid, but in step 2 in x_3 we obtain false answers, because the speed is not the same as the user prompt. The problem was largely with understanding speed. So, we need to use a regular expression (REGEX) method [20]. It means the algorithm ignores “st”, “nd”, “rd”, “th” and from Hungarian ordinal number expression “-es”, “-as”, “-ös”, “-os”.

With this algorithm, the prompts in Table 1 are rechecked and the system returns the correct values, registers, and speed.

This method runs after every user prompt, which causes slow answers and execution. Our algorithm is made for the whole system. We have seven storage units, and the speed is configurable between 1–100.

There are 693 tasks to be executed. Total tests: 693. Passed: 693. Failed: 0. The model is reliable. The Correction Rate is 1.

This test uses a big GPU, so the running time of the test depends on the GPU. If the hardware is as much as possible, the running time will decrease. If ran once, we do not need to use it again before the user prompts. The current prompt set is structured but future testing will include free form or informal expressions.

6. Conclusions

One of the challenges is the significant energy and storage requirements of an LLM [21], which are considerably higher than those of a traditional generative model. Ensuring the efficient operation of such a system requires sufficient hardware and software resources. Additionally, proper tokenization is crucial, as it optimizes data processing and reduces computational demands. Selecting the best model for this task is essential, considering factors such as accuracy, performance, and response time.

In natural language-based control systems, both the user’s language and linguistic structure are critical factors. There are major structural differences between English and Hungarian, so the system must be capable of effectively handling both languages. Specifically for Hungarian, a processing program that can accurately recognize and interpret complex structures and unique expressions must be developed. Inflected words, conjugation, and ambiguous phrases pose significant challenges for an LLM. Therefore, an algorithm must be designed to accommodate the unique characteristics of the Hungarian language and ensure precise interpretation of user commands. If the language processing functions correctly, controlling a PLC through natural language will become smoother and more efficient. The current LLM model handles Hungarian words to some extent, but adjective-forming suffixes require separate processing [22,23]. Future research could focus on optimizing prompt formulations across multiple languages to ensure consistent execution efficiency, particularly in systems involving multilingual data handling. Investigating how different language structures, such as agglutinative vs. analytic languages, impact execution performance could uncover model-specific inefficiencies and could focus on voice-based user prompt in Hungarian.

Voice in Hungarian:

I asked the model: “Vidd a 2-es tárhelyre 40%-al”,

Write the prompt with hand (h) or with voice(b)? [h/v]: V

Say what you want (etc. ‘set storage one to 35 percent’):

Recognize text: need a catfish tare nai,

Author Contributions: Conceptualization, O.H.; methodology, O.H. and M.K.; software, M.K.; validation, M.K., formal analysis, O.H. and M.K.; investigation, O.H.; resources, M.K.; data curation, O.H.; writing—original draft preparation, M.K.; writing—review and editing, O.H.; visualization, M.K.; supervision, O.H.; project administration, M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Washfy, T.M.; Noor, A.K. Rule-based natural-language interface for virtual environments. *Adv. Eng. Softw.* **2002**, *33*, 155–168. [CrossRef]
2. Pillai, A.S. Advancements in natural language processing for automotive virtual assistants enhancing user experience and safety. *J. Comput. Intell. Robot.* **2023**, *3*, 27–36.
3. Zhao, W.X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. A survey of large language models. *arXiv* **2023**, arXiv:2303.18223. [CrossRef] [PubMed]
4. Wu, T.; He, S.; Liu, J.; Sun, S.; Liu, K.; Han, Q.L.; Tang, Y. A brief overview of ChatGPT: The history, status quo and potential future development. *IEEE/CAA J. Autom. Sin.* **2023**, *10*, 1122–1136. [CrossRef]
5. Introducing LLaMA: A Foundational, 65-Billion-Parameter Large Language Model. Available online: <https://ai.meta.com/blog/large-language-model-llama-meta-ai/> (accessed on 12 June 2025).
6. Gao, T.; Fisch, A.; Chen, D. Making pre-trained language models better few-shot learners. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Online, 1–6 August 2021; pp. 3816–3830. [CrossRef]
7. Taylor, J.H.; Stringer, P.G. An autosynthesizing non-linear control system using a rule-based expert system. *Int. J. Adapt. Control Signal Process.* **1991**, *5*, 21–39. [CrossRef]
8. Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.; Lacroix, T.; Roziere, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. LLaMA: Open and efficient foundation language models. *arXiv* **2023**, arXiv:2302.13971. [CrossRef]
9. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS 22), New Orleans, LA, USA, 28 November–9 December 2022.
10. Servantez, S.; Barrow, J.; Hammond, K.; Jain, R. Chain of logic: Rule-based reasoning with large language models. *Findings Assoc. Comput. Linguist. ACL* **2024**, *2024*, 2721–2733. [CrossRef]
11. Ye, Q.; Axmed, M.; Pryzant, R.; Khani, F. Prompt engineering a prompt engineer. *arXiv* **2024**, arXiv:2311.05661. [CrossRef]
12. Lo, L.S. The art and science of prompt engineering: A new literacy in the information age. *Internet Ref. Serv. Q.* **2023**, *27*, 203–210. [CrossRef]
13. Xia, Y.; Jazdi, N.; Zhang, J.; Shah, C.; Weyrich, M. Control industrial automation system with large language models. *arXiv* **2024**, arXiv:2409.18009. [CrossRef]
14. Ioannides, M.G. Design and implementation of PLC-based monitoring control system for induction motor. *IEEE Trans. Energy Convers.* **2004**, *19*, 469–476. [CrossRef]
15. Saleh, M.S.; Mohammed, K.G.; Al-sagar, Z.S.; Sameen, A.Z. Design and implementation of PLC-based monitoring and sequence controller system. *J. Adv. Res. Dyn. Control Syst.* **2018**, *10*, 2281–2289.
16. Qin, C.; Zhang, A.; Zhang, Z.; Chen, J.; Yasunaga, M.; Yang, D. Is ChatGPT a general-purpose natural language processing task solver? *arXiv* **2023**, arXiv:2302.06476. [CrossRef]
17. Trifan, M.; Ionescu, B.; Ionescu, D. A real time self-generating control for AI platforms. In Proceedings of the IEEE 18th International Symposium on Applied Computational Intelligence and Informatics (SACI), Siófok, Hungary, 21–25 May 2024. [CrossRef]
18. JSON—Introduction. Available online: https://www.w3schools.com/js/js_json_intro.asp (accessed on 12 June 2025).
19. Kochmar, E. *Getting Started with Natural Language Processing*; Manning Publications: Shelter Island, NY, USA, 2022; ISBN 9781617296765.
20. Chen, Q.; Banerjee, A.; Demiralp, C.; Durrett, G.; Dillig, I. Data extraction via semantic regular expression synthesis. *Proc. ACM Program. Lang.* **2023**, *7*, 1848–1877. [CrossRef]
21. Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T.B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; Amodei, D. Scaling laws for neural language models. *arXiv* **2020**, arXiv:2001.08361. [CrossRef]

22. Dasgupta, I.; Lampinen, A.K.; Chan, S.C.Y.; Creswell, A.; Kumaran, D.; McClelland, J.L.; Hill, F. Language models show human-like content effects on reasoning. *arXiv* **2022**, arXiv:2207.07051. [[CrossRef](#)]
23. Hoffmann, J.; Borgeaud, S.; Mensch, A.; Buchatskaya, E.; Cai, T.; Rutherford, E.; Casas, D.D.L.; Hendricks, L.A.; Welbl, J.; Clark, A.; et al. Training compute-optimal large language models. *arXiv* **2022**, arXiv:2303.15556. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.