Enhancing Information Extraction for Simulation Modelling Using Large Language Models

Marco Brunschwiler*, Lukas Hollenstein, Dominik Kunz, Samuel Wehrli

ZHAW Zurich University of Applied Science, Institute of Computational Life Science, Wädenswil, Switzerland
*marco.brunschwiler@zhaw.ch

SNE 36(3), 2026, 123-129, DOI: 10.11128/sne.36.tn.10782
Selected ASIM SPL 2025 Postconf. Publication: 2026-02-15
Received Revised Extended: 2026-05-27; Accepted: 2026-07-20
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. This study investigates the application of Large Language Models (LLMs) for structured information extraction in the context of simulation modelling. Specifically, the performance of Meta-Llama-3-70B-Instruct was evaluated in extracting simulation model components — such as sources, process steps, and parameters — from unstructured textual descriptions of logistics systems.

The evaluation compared zero-shot and few-shot prompting strategies and analysed model consistency across 50 repeated runs per prompt configuration. Metrics such as precision, recall, and F1-score were computed using fuzzy string matching. Results show that few-shot prompting improves extraction performance overall, with statistically significant gains in categories such as model parameters and sources.

However, inconsistent effects were observed for other categories, including a decrease in key metrics identification. Variability analysis using the coefficient of variation revealed differing levels of output stability across categories and prompts.

The findings indicate that few-shot prompting with carefully chosen examples enhances model reliability but may introduce risks of hallucinated outputs and prompt-specific overfitting. Recommendations are provided for prompt engineering in simulation modelling applications. Future work will consider other LLM architectures, dynamic prompt selection, and integration of external validation mechanisms.

Introduction

Simulation modelling is a cornerstone in analysing and optimising complex systems, yet practitioners frequently face challenges when translating unstructured or inadequately structured information and data into coherent formal model representations [2]. Traditionally, practitioners in simulation modelling invest considerable manual effort to meticulously extract crucial model elements from extensive textual descriptions. This manual transformation process is time-consuming, labour-intensive, and prone to inconsistencies due to subjective interpretation and varying levels of expertise [10].

Structured information extraction from unstructured text has long been a central challenge in natural language processing. Traditional approaches—such as rule-based systems, pattern matching, and supervised machine learning—sought to automate this task by encoding expert knowledge or training models on large, labelled datasets [4]. However, these methods typically require significant domain expertise, extensive manual rule creation, or substantial annotated data. Consequently, they tend to lack adaptability, struggle to generalise across domains, and often require considerable recalibration to remain effective in new or evolving contexts [4, 10].

The emergence of Large Language Models (LLMs) marks a transformative shift in the field of natural language processing. Pre-trained on massive text corpora, LLMs demonstrate remarkable capabilities in understanding context, generating human-like text, and performing complex tasks with minimal domain-specific training, often through in-context learning or fine-tuning [8, 11].

Recent research has increasingly explored their potential for structured data extraction across various domains. For example, the "ExtractGPT" approach shows that LLMs can extract product attributes from textual descriptions with limited training data, achieving competitive performance [3]. Similarly, fine-tuned LLMs have been applied to complex scientific texts, highlighting their versatility and robustness in extracting structured information from unstructured input [4].

These capabilities are particularly valuable in domains such as Model-Driven Engineering (MDE) and Discrete-Event Simulation (DES), where formalizing textual system descriptions is a critical step in model construction [17].

Prompt engineering has emerged as a critical factor in optimising LLM performance [15]. The way a task is presented to an LLM — including the instructions, the inclusion of examples (few-shot prompting), and the overall prompt structure — can significantly influence output quality and accuracy. White et al. [15] compiled a catalogue of prompt patterns, offering a systematic approach to designing effective prompts for various tasks, including information extraction.

In the field of simulation modelling, the use of Artificial Intelligence (AI) and Large Language Models (LLMs) is steadily increasing. For example, AI has been applied to generate syntactically and functionally valid simulation models of logistics systems from natural language descriptions in the form of Python code [8]. However, these approaches often lack systematic evaluation regarding the quality and robustness of the generated models. Similarly, LLMs have been used in multi-agent systems to support model parameterisation within Digital Twins, suggesting their potential to streamline simulation workflows [16].

Nevertheless, while these developments point to a broader applicability of LLMs in simulation modelling, detailed investigations into the extraction of model components—such as sources, processes, parameters, and performance metrics—from textual descriptions remain limited.

This study investigates the capabilities of a state-of-the-art LLM, specifically Meta-Llama-3-70B-Instruct [1, 6], in extracting model elements from natural-language descriptions of logistic systems, thought of as a first step towards deriving model specifications from unstructured system information.

It aims to answer three main questions:

1. How accurately can the LLM identify and extract components of simulation models from various texts that describe logistics systems?
2. How do different prompting strategies—ranging from zero-shot to few-shot with illustrative examples—influence extraction quality?
3. How consistent are the LLM outputs across repeated executions of identical prompts, and which implications thereof follow the model's inherent robustness and reliability for practical applications?

To this end, the influence of different prompting strategies is evaluated systematically, applying fuzzy string matching [5] based on Levenshtein distance [9] and qualitatively using semantic similarity assessment [12], along with standard information retrieval metrics (precision, recall, and F1-score) to provide a quantitative measure of performance.

Finally, the variability across repeated executions is analysed to evaluate robustness.

The remainder of this paper is structured as follows: Section 1 details the methodology, including the experimental design, dataset characteristics, prompt variations, and the specific evaluation metrics used. Section 2 presents and interprets the results of the experiments. Finally, Section 3 summarises the key findings and contributions and concludes with implications and future research directions.

1 Methodology

To rigorously evaluate the capabilities of LLMs in extracting structured information pertinent to simulation modelling, a comprehensive experimental methodology was designed and executed.

This section details the specific LLM employed, the dataset constructed for the evaluation, the experimental procedures followed, the targeted output format, and the multifaceted evaluation metrics used to assess performance and robustness. Prompt examples, evaluation scripts, and replication materials are available on GitHub: https://github.zhaw.ch/SimOpt/Enhancing-Information-Extraction-for-Simulation-Modelling-Using-Large-Language-Models.git

1.1 Large Language Model

The core of this investigation utilised the Meta-Llama-3-70B-Instruct model, a state-of-the-art LLM developed by Meta AI [1, 6].

This model was selected due to its advanced instruction-following capabilities and strong performance benchmarks across various natural language processing tasks, making it a suitable candidate for the complex information extraction required in the context of simulation modelling.

At the time of this study, the model was freely accessible via the Groq platform [7] and offered unlimited inquiries, thereby enabling extensive experimentation without resource constraints.

1.2 Dataset and Task Description

The dataset comprised five (5) distinct textual descriptions detailing logistics systems. These descriptions were carefully curated to represent varying levels of complexity (Tab. 1). For each textual description, a corresponding structured list of expected model elements was manually created by domain experts, serving as the ground truth for evaluation.

In this context, system complexity refers to the structural and functional richness of the described logistics processes. While simple systems primarily consist of linear sequences of process steps, medium-complexity systems include additional resource interactions, and complex systems further incorporate elements such as storage stages and pull-based control mechanisms.

The task presented to the LLM was to process each textual description and extract relevant model elements, organizing them into predefined categories.

System	Complexity	Description of complexity
Ice cream production	Simple	
Service operations	Simple	Linear sequence of process steps
Laboratory process	Simple	
Restaurant service	Medium	Includes additional resource interactions
Juice factory	Complex	incorporates elements such as storage stages and pull-based control mechanisms

Table 1: Logistics system and their respective levels of complexity.

These categories included the essential simulation components:

- Sources (e.g., delivery of raw ingredients)
- Process steps (e.g., washing, grinding, heating)
- Sinks (e.g., shipping of products)
- Parameters (e.g., process time, arrival rate)
- Decision variables (e.g. capacities, number of employees)
- Key metrics (e.g., queue length, waiting time, utilisation)

1.3 Experimental Setup

The experimental design focused on assessing the impact of different prompting strategies on the LLM's extraction performance.

Each of the five system descriptions was presented to the Meta-Llama-3-70B-Instruct model using six (6) distinct prompt configurations. One configuration represented a zero-shot prompt, where the model received only the task instruction and the target system description without any illustrative examples.

The remaining five configurations employed a few-shot learning approach. In these configurations, the prompt included the task instruction, the target system description, and one complete example consisting of one of the other system descriptions paired with its corresponding ground truth output in JavaScript Object Notation (JSON).

To evaluate the consistency and reliability of the model's output, each of the distinct 25 scenarios (i.e., all combinations of system description and prompt configurations) was executed 50 times (replications), yielding a substantial dataset for statistical analysis.

Welch's t-test (at 99% confidence level) was employed to test for significant differences of the mean performance with respect to different prompts and the Kruskal-Wallis test followed by a Wilcoxon post-hoc test (at 99% confidence level) were used to test for significant differences between few-shot prompts with examples of different complexity.

For the experiments and the analysis Python (version 3.11.4) was used as well as the packages SciPy (version 1.11.2) [14] and Pingouin (version 0.5.5) [13].

1.4 Output Format

To facilitate automated evaluation and ensure structured output, the LLM was explicitly instructed to format its response as a single, nested JSON object. This JSON object was required to contain keys corresponding to the predefined model element categories, with the values being lists of strings representing the extracted elements for each category. Adherence to this structured JSON format was crucial for the subsequent automated evaluation pipeline.

1.5 Evaluation Metrics

A multi-pronged approach was adopted to evaluate the quality and consistency of the LLM's extractions, leveraging both textual similarity and standard information retrieval metrics.

Textual and Semantic Similarity

Recognizing that LLM outputs might capture the correct semantic meaning with slight variations in phrasing, two similarity measures were employed. Firstly, fuzzy string matching was performed using the Fuzzywuzzy library [5], which uses the Levenshtein distance [9] between the predicted and expected strings. A threshold of 50% on the similarity was used to determine if a predicted item was considered a match to an expected item, allowing for minor textual discrepancies.

Additionally, semantic similarity was qualitatively assessed using sentence transformers based on Siamese BERT-Networks [12]. This involved generating vector embeddings for both expected and predicted items and calculating the cosine similarity between these embeddings, providing a measure of semantic equivalence independent of exact wording.

Information Retrieval Metrics

Based on the matches identified through the similarity measures, standard information retrieval metrics were calculated for each category and overall.

True positives (TP) were counted when an expected item was correctly identified (found a similar item in the prediction). False negatives (FN) were counted when an expected item was missed by the model. False positives (FP) were counted when the model predicted an item that was not present in the expected list (hallucination).

From these counts, the following metrics were derived:

- **Precision = TP / (TP + FP)**: Fraction of predicted items correctly identified.
- **Recall = TP / (TP + FN)**: Fraction of expected items correctly identified.
- **F1-score = (2 × precision × recall) / (precision + recall)**: The harmonic mean of precision and recall, providing a balanced measure of accuracy.

These metrics were calculated for individual categories first, then at category level as well (missing categories increase FN, hallucinated categories increase FP), and finally aggregated across all categories to yield an overall score of the model's performance. The analysis focused on the F1-score.

1.6 Consistency Analysis

To evaluate the robustness and reliability of the LLM's output across multiple runs, the consistency of the performance metrics was analysed. The Coefficient of Variation (CV), defined as the standard deviation divided by the mean, was computed for the metrics defined above for each scenario (combination of system description and prompt configuration) across 50 runs. A lower CV indicates greater stability and more consistent performance from the model under identical conditions.

2 Results

Each of the 25 scenarios (5 system descriptions times 5 prompts) were executed 50 times using the Meta-Llama-3-70B-Instruct model. The responses were evaluated as described in Sec. 1.5 to yield an F1-score for each of the six model element categories (see Section 1.2) and aggregated as an overall score. Some runs were excluded from the analysis due to formatting issues: 2 of 250 zero-shot scenarios and 12 out of 1000 few-shot scenarios.

2.1 Comparison of Zero-shot and Few-shot Prompting

The impact of the general prompting strategy on model performance was evaluated comparing between zero-shot (instruction only) and few-shot (instruction plus example) scenarios, see Table 2.

Category	Few-shot			Zero-shot			p-value
	count	mean	std	count	mean	std	
Overall	7729	**0.528**	0.466	2032	0.462	0.47	$< 10^{-7}$
Sources	988	**0.284**	0.451	248	0.202	0.402	0.00509
Process steps	988	0.564	0.361	248	0.592	0.412	0.32595
Sinks	988	0.240	0.425	248	0.202	0.402	0.17915
Decision variables	988	0.968	0.171	248	**1**	0	$< 10^{-7}$
Model parameters	988	**0.898**	0.236	248	0.297	0.386	$< 10^{-7}$
Key metrics	932	0.322	0.467	248	**0.605**	0.49	$< 10^{-7}$

Table 2: Comparison of F1-scores between few-shot and zero-shot prompting across model element categories and overall. P-values are given from Welch's t-test.

In the overall F1-score, few-shot prompting led to a significant improvement compared to zero-shot prompting. However, individual model element categories showed varying results: The most substantial gain was observed for model parameters. Sources also showed a significant, though smaller, improvement. In contrast, key metrics exhibited a significant performance drop under few-shot prompting. Process steps and sinks showed no statistically significant differences. Decision variables remained near-perfect in both settings, with only a minimal decline in the few-shot condition.

2.2 Prompt Example Complexity and Generalisation

Focusing on the few-shot prompt scenarios, the overall F1-score for different complexity groups of the presented examples (see Table 1) was compared.

The Kruskal-Wallis test showed significant differences ($p < 10^{-18}$) between the complexity groups, while the Wilcoxon post-hoc test revealed that simple examples (mean F1 of 0.565) outperformed medium complex (mean F1 of 0.487) as well as complex examples (mean F1 of 0.472), both significantly ($p < 10^{-8}$).

Medium complex and complex examples did not lead to significantly different F1-scores ($p = 0.091$).

The analysis revealed several hallucinated outputs (i.e., false positives) that, while not listed in the summary table, adversely affected overall performance.

These hallucinations involved fabricated or irrelevant category labels, such as:

- "simulation of ice cream production"
- "distribution centre" (not existing in the ground truth)
- "inbound activities" (created as an incorrect category header)

2.3 Consistency and Stability of Model Outputs

To assess the robustness of model performance the coefficient of variation (CV) of the overall F1-score was calculated over all 50 replications for each of the 25 scenarios, see Figure 1.

The CV provides a normalised measure of variability that enables comparison across categories.

The results reveal clear patterns of variability across the different categories. In the sources category, variability increased when examples were included in the prompt, indicating greater inconsistency in those cases.

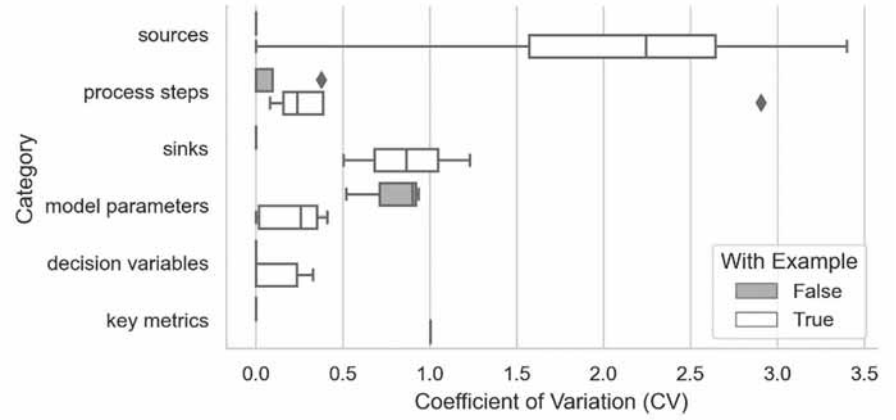

Figure 1: Category-specific variability – coefficients of variation of the F1-score by category and example-use summarising n=5 zero-shot scenarios (without example) and n=20 few-shot scenarios (with example).

A similar trend was found for process steps and key metrics, where the inclusion of examples also led to higher variability. In contrast, model parameters showed a decrease in variability under the with-example condition, suggesting more consistent performance. For decision variables, variability remained very low in both conditions. Sinks, on the other hand, exhibited moderate variability when examples were present, but showed no variability without examples. Overall, the CV analysis highlights that the degree of output variability differs notably by model element category and is not uniformly reduced nor increased through the use of examples.

Importantly, variability also depended on the specific example provided in the few-shot prompt. Providing different examples influenced the consistency of outputs distinctly across the evaluated categories, highlighting the sensitivity of model performance to the selection of specific prompts. In some cases, providing an example lowered the observed variability. However, certain categories showed significantly higher variability, thus, less consistent extractions.

In general, categories that are well-structured and clearly defined tended to produce more stable outputs, whereas those involving higher complexity or ambiguity yielded greater dispersion in results. A detailed analysis of these findings is outside the scope of this work and may need more scenarios.

3 Discussion and Conclusions

The results demonstrate that prompting strategies affect extraction performance in a category-specific manner. Few-shot prompting notably improved performance for model parameters and sources, indicating that structured examples help the model interpret and extract these more context-sensitive elements.

The substantial gain in F1-score for model parameters suggests the model relies heavily on format cues to identify variable-like content.

In contrast, performance for sinks and process steps remained stable across prompting strategies. These elements likely occur more explicitly in natural language, making them easier to be identified even in zero-shot settings.

Interestingly, key metrics performed significantly worse under few-shot prompting, suggesting that the examples may have introduced ambiguity or misled the model regarding the structure of performance indicators.

This highlights a potential risk of overfitting or misgeneralisation when examples are poorly chosen. Decision variables, although showing a small decline, remained near perfect in both conditions, potentially indicating a ceiling effect or unnecessary complexity introduced by few-shot prompts.

Prompt complexity also played a notable role. Simple examples led to better generalisation, while prompts based on medium or complex systems reduced performance. This suggests that simpler inputs help the model generalise better and minimise the cognitive load on the model.

Variability analysis showed that prompting with examples does not uniformly enhance output stability. In categories like model parameters, consistency improved, but others—such as sources and key metrics—exhibited greater variability. These effects were also influenced by the specific example used in the prompt, emphasizing that prompt selection matters not just for accuracy but also for robustness.

Overall, prompts with structurally clear, well-defined elements produced more stable results. Categories that were more ambiguous or complex in nature led to increased dispersion. These findings suggest that prompt design should prioritise clarity and simplicity, tailored to the target extraction category.

Building on these insights, future research could explore adaptive prompt strategies, such as dynamic example selection based on input similarity or prompt refinement guided by reinforcement learning. Furthermore, integrating external knowledge sources or leveraging chain-of-thought prompting may enhance recall for challenging categories, such as source elements.

Comparative studies across different LLM families could reveal whether the effects of prompt complexity generalise across architectures. In addition, evaluating a broader set of example systems and test cases—including more complex scenarios—would offer deeper insight into how LLMs can support practitioners throughout the simulation modelling process.

Finally, linking information extraction with code generation represents a logical next step towards fully integrated simulation development environments.

The investigation at hand underscores the efficacy of few-shot prompting with simple, well-structured examples for enhancing LLM information extraction. By systematically quantifying performance gains, variability reductions, and error profiles, actionable guidelines for prompt engineers were provided.

These may set the stage for next-generation adaptive prompting frameworks.

While the findings provide a solid foundation for improving information extraction in simulation modelling, a practical application in real-world simulation projects—where domain heterogeneity, incomplete descriptions, and complex system behaviours prevail—remains a significant step ahead and will require further validation, tooling, and integration into modelling workflows.

Acknowledgement

This work resulted from a project funded by ZHAW digital through the Digital Futures Fund for R&D.

Publication Remark

This contribution is the revised version of the conference version for **ASIM SPL 2025** - 21. ASIM Fachtagung Simulation in Produktion und Logistik - Dresden 2025, published in *Simulation in Produktion und Logistik 2025*, ASIM-Mitteilung Nr. 194, ISBN 978-3-86780-806-4, e-ISBN 978-3-86780-809-5, Vol. DOI 10.25368/2025.233, Article DOI 10.25368/2025.282

References

[1] AI@Meta: Llama 3 model_card. 2024. https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md, accessed January 31st, 2025.

[2] Benedettini O, Tjahjono B. Towards an improved tool to facilitate simulation modelling of complex manufacturing systems. International Journal of Advanced Manufacturing Technology 43 (2009), pp. 191–199.

[3] Brinkmann A, Shraga R, Bizer C. ExtractGPT. Exploring the potential of large language models for product attribute value extraction. In Haghighi P, Greguš M, Kotsis G, Khalil I. (Eds.): Information integration and web intelligence. Cham: Springer Nature Switzerland 2025, pp. 38–52.

[4] Dunn A, Dagdelen J, Walker N, Lee S, Rosen AS, et al. Structured information extraction from complex scientific text with fine-tuned large language models. arXiv, 2022. http://arxiv.org/abs/2212.05238, accessed January 21st, 2025.

[5] fuzzywuzzy: Fuzzy string matching in Python. 2020. https://github.com/seatgeek/fuzzywuzzy, accessed January 24th, 2025.

[6] Grattafiori A, Dubey A, Jauhri A, Pandey A, Kadian A, et al. The Llama 3 herd of models. arXiv, 2024. http://arxiv.org/abs/2407.21783, accessed January 24th, 2025.

[7] Groq: Groq is fast AI inference. 2023. https://groq.com/, accessed May 15th, 2025.

[8] Jackson I, Jesus Saenz M, IvanovD. From natural language to simulations: applying AI to automate simulation modelling of logistics systems. International Journal of Production Research 62 (2024), pp. 1434–1457.

[9] Levenshtein VI. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 10 (1966), pp. 707.

[10] Li Y, Ji W. AbouRizk SM. Automated abstraction of operation processes from unstructured text for simulation modeling. In: Proc. 2020 Winter Simulation Conference (WSC), Orlando (USA), December 2020. IEEE 2020, pp. 2517–2525. https://ieeexplore.ieee.org/document/9383953/, accessed January 24th, 2025.

[11] May MC, Neidhöfer J, Körner T, Schäfer L, Lanza G. Applying natural language processing in manufacturing. Procedia CIRP 115 (2022), pp. 184–189.

[12] Reimers N, Gurevych I: Sentence-BERT: sentence embeddings using Siamese BERT-networks. In: Proc. 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong (China), Nov. 3–7, 2019. Association for Computational Linguistics 2019, pp. 3980–3990. https://www.aclweb.org/anthology/D19-1410, accessed January 29th, 2025.

[13] Vallat R. Pingouin: statistics in Python. Journal of Open Source Software 3 (2018), p. 1026.

[14] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods 17 (2020), pp. 261–272.

[15] White J, Fu Q, Hays S, Sandborn M, Olea C, et al. A prompt pattern catalog to enhance prompt engineering with ChatGPT. arXiv, 2023. http://arxiv.org/abs/2302.11382, accessed January 24th, 2025.

[16] Xia Y, Dittler D, Jazdi N, Chen H, Weyrich M. LLM experiments with simulation: large language model multi-agent system for simulation model parametrization in digital twins. In: Proc. of the 2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA), 2024, pp. 1–4. https://ieeexplore.ieee.org/abstract/document/10710900, accessed January 28th, 2025.

[17] Xia Y, Shenoy M, Jazdi N, Weyrich, M. Towards autonomous system: flexible modular production system enhanced with large language model agents. arXiv, 2023. http://arxiv.org/abs/2304.14721, accessed September 11th, 2023.