Systems & Control Transactions

Research Article - Peer Reviewed Conference Proceeding
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale,
Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

Application of Artificial Intelligence in process simulation tool

Nikhil Rajeev[a], Suresh Jayaraman[a*], Prajnan Das[b], Srividya Varada[a]

[a] AVEVA Group Ltd, United States of America
[b] Cognizant Technology Solutions U.S. Corporation, United States of America
* Corresponding Author: suresh.jayaraman@aveva.com

ABSTRACT

Process engineers in the Chemical and Oil & Gas industries extensively use process simulation for the design, development, analysis, and optimization of complex systems. This study investigates the integration of Artificial Intelligence (AI) with AVEVA™ Process Simulation (APS), a next-generation commercial simulation tool. We propose a framework for a custom chatbot application designed to assist engineers in developing and troubleshooting simulations. This chatbot application utilizes a custom-trained model to transform engineer prompts into standardized queries, facilitating access to essential information from APS. The chatbot extracts critical data regarding solvers and thermodynamic models directly from APS to help engineers develop and troubleshoot process simulations. Furthermore, we compare the performance of our custom model against OpenAI technology. Our findings indicate that this integration significantly enhances the usability of process simulation tools, promoting more innovative and cost-effective engineering solutions.

Keywords: Artificial Intelligence, Process Design, Machine Learning, Chatbot

INTRODUCTION

Process simulation is a model-based representation of chemical, physical, and biological processes to analyze, optimize, and design. Process simulation tools are widely used in Chemical, and Oil & Gas industries in various phases of a project. Process simulation can enable the transformation of process engineering as part of the digital twin, an important building block of digital transformation [1]. Simulation and modeling can help accelerate the design cycle of chemical processes. The rapid evolution of Artificial intelligence (AI) and Machine Learning (ML) technologies has significant implications across various domains including engineering. AI and ML technologies have been widely used in various applications of chemical engineering such as process modeling, simulation, control, classification, fault detection and diagnosis [2]. Recent research has focused on integrating AI with process simulation tools to enhance design efficiency [3]. Stergiou et al discuss combining ML algorithms with process simulation tool to optimize plant design parameters, resulting in more efficient and cost-effective chemical production [4]. The application of AI techniques such as Artificial Neural Networks (ANN) is used to model the operation of hydrogen production system [5].

A chatbot is a computer-based program that simulates human conversation with an end user, and they are usually built using AI and Natural Language Processing (NLP) with the goal of interpreting human questions and automating customized responses [6]. ChatGPT (Chat Generative Pre-Trained Transformer) is a recently developed conversational chatbot created by OpenAI [7], and it uses Natural Language Processing (NLP) to generate human-like responses to user input.

This research paper discusses applications of AI in a next-generation process simulation tool using both custom trained model and OpenAI model to answer questions based on the process simulation. The performance of these models is quantified and compared using different types of similarity scores. This innovative approach of integrating AI technology with a next generation process simulation tool is expected to transform AVEVA™ Process Simulation (APS) by making it more user-friendly and efficient, leading to novel and cost-effective solutions in process engineering.

METHODOLOGY

The architecture of a chatbot encompasses its foundational structure and design principles. It outlines how the chatbot interprets and processes text, influenced by various factors including the domain, use case, type of chatbot, and prompts used [8]. Models in ChatBot utilize NLP engines to process user input and transform it into machine-readable formats; however, traditional NLP engines struggle to fully comprehend human language. To address this limitation, large language models (LLM) were developed to enhance AI's understanding of natural language, enabling the generation of more realistic and coherent human-like text.

Chatbot architecture

The architecture for implementing the chatbot is illustrated in Figure 1. User input is processed by either a LLM or NLP, which may be a custom-trained model or a pre-trained one, such as OpenAI's Generative Pre-Trained Transformer (GPT) or Google's BERT, LaMDA, and PaLM [9]. The chatbot leverages the LLM to extract relevant information from unstructured textual data through techniques like text classification and entity recognition.

Text classification is essential for identifying user intents by categorizing messages into classes such as inquiries or requests, which guides the chatbot's response strategy. Entity recognition enhances this process by extracting specific information, such as names or dates, from user inputs. Together, these features enable the chatbot to engage in more meaningful, context-aware interactions, significantly improving user satisfaction and overall effectiveness. The unstructured text is then converted into a structured XML format based on user input, which is utilized to build simulations or troubleshoot existing ones. The Next generation process simulation software, AVEVA™ Process Simulation (APS) developed by AVEVA, is used in this work. APS is designed for modeling, simulating, studying thermodynamic behavior and optimizing process operations in industries such as oil and gas, chemicals, and manufacturing.

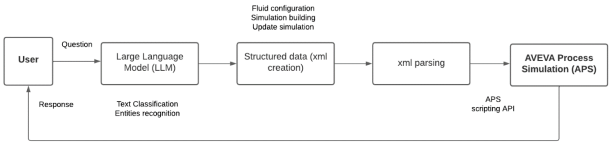

Figure 1. Architecture for ChatBot implementation.

The chatbot framework interfaces with the APS through the scripting interface of the process simulator tool, enabling bidirectional communication between the process simulation and the chatbot application using Application Programming Interface (API's) in Python. These API's facilitates tasks such as creating a component slate with thermodynamic models, building simulations, and extracting essential solver and thermodynamic information from the simulation results to help process engineers design, analyze and optimize chemical processes.

Custom trained model

SpaCy is a Python-based open-source NLP library used to develop custom models. It features built-in pre-trained pipelines and supports tokenization, as well as neural network models for tagging, parsing, named entity recognition (NER), and text classification. In our custom model, text classification and NER are integrated within the pipeline, sharing a token-to-vector layer between the two components. The text classification component categorizes user input into specific types, such as "sim building," "model query," or "fluid building," based on labeled training text. Meanwhile, the NER component identifies and classifies named entities in the text into predefined categories, including models, component names, and thermodynamic methods.

To generate training data for the spaCy model, we utilize Faker, a Python package that creates fake data. The training data patterns are designed to mimic various likely user inputs, and entities are annotated using spaCy's rule-based matching as shown in Figure 2.

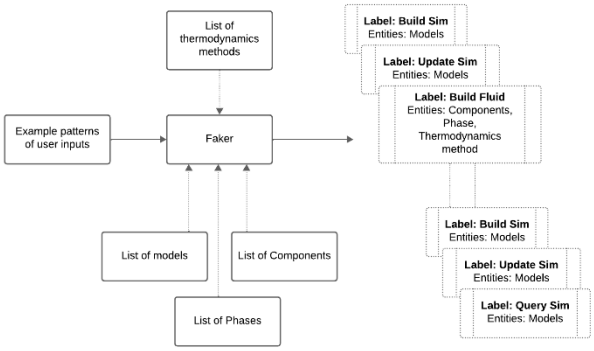

Figure 2. Preparation of training data.

The custom training of the spaCy pipeline components is saved as the best model on disk. A simple scoring system, applying equal weights (50-50) to text classification and NER, determines the best model. Once constructed, the NLP pipeline utilizes the pretrained components, allowing user input to be fed into the spaCy pipeline for classification and custom entity identification. Based on this classification, the appropriate XML generation occurs, with named entities represented as XML tags and attributes. From this stage of XML generation onward, the code is consistent between the OpenAI-based model and the custom model. The XML data is then parsed, and the information is sent to the APS through a scripting interface for feedback. Standard

XML template used are presented in the supplementary material.

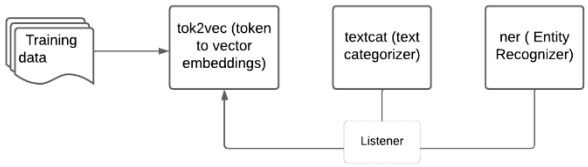

Figure 3. SpaCy training pipeline components.

Pre-trained model

The pretrained GPT-3.5-turbo model is utilized as the base model through the Python API endpoint. Communication with the OpenAI model is facilitated by sending HTTP requests to the API endpoints and receiving corresponding responses.

RESULTS AND DISCUSSION

To compare the XML results generated by OpenAI and the custom model, it is essential to assess both the structural and content-based similarities. The most effective approach to evaluating the response is by examining both the structural and contextual aspects of the returned XML, such as the similarity of tags and the accuracy of the extracted data. This can be accomplished by establishing a "golden file," which represents the desired XML response, and then comparing the generated XML against this reference. Four metrics have been utilized to assess these differences, each focusing on a distinct aspect of the generated XML file.

Jaccard similarity

Jaccard similarity quantifies the overlap between two sets. When applied to XML, focusing solely on tags, it evaluates the similarity of the structures between two XML documents. This metric is useful for assessing how closely the structure of a generated XML document aligns with the desired structure. The process involves extracting the unique tags from both XML documents, converting them into sets, and computing the Jaccard similarity, as defined by [10]:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A|+|B|-|A \cap B|} \quad (1)$$

By evaluating both the structural elements of the XML and the data they contain, it enables the assessment of the correctness of the XML structure as well as the accuracy of the data.

Cosine Similarity

Cosine similarity is employed to assess the similarity between two documents by calculating the cosine value between the term vectors of the documents [11]. This method transforms documents into vectors within a multi-dimensional space, enabling effective comparison of textual data extracted from XML elements. Consequently, cosine similarity is particularly well-suited for evaluating the semantic similarity of text within tags.

Levenshtein Distance

Levenshtein Distance measures the differences between two sequences, accounting for variations in both equal-length and unequal-length sequences [12]. It determines the minimum number of single-character edits necessary to transform one string into another to identify even the smallest textual differences between two files.

Case studies

The ChatBot framework is evaluated through a series of case studies designed to simulate real-time user prompts typically posed by process engineers in an industrial setting as presented in supplemented material. The intent, along with the user prompts and associated metrics, is presented in the following section.

Case study 1

Intent Build a simulation in APS using user prompts in a sentence structure with intentional grammatical errors and typos.

User Prompts

Prompt 1: Add a source, compressor, valve, pump, valve and sink and connect them. Now update temperature and pressure conditions of source model to 450 K and 250 psi

Prompt 2: Try connecting source, compressor, valve, pump, valve and sink. Also update temperature and pressure of source model to 450 K and 250 psi

Prompt 3: Try connecting source, compressor, valve, pump, valve and sink. Also update T and P of source model to 450 K and 250 psi

Prompt 4: Try connecting source, compressor, valve, pump, valve and sink. Also update temperature and pressure of source model to 100 C and 250 psi1

Prompt 5: Build a simulation with source, valve and sink. Also update temperature and pressure of source model to 100 C and 250 psi

Table 1: Metrics for xml generation from user prompts using pre-trained ChatGPT model

Prompt#	Jaccard Similarity		Cosine Similarity	Levenshtein Distance
	Tags	Data		
1	1	0.65	0.99	6
2	0.63	0.51	0.99	336
3	0.63	0.51	0.99	336
4	1	0.65	0.99	6
5	0.34	0.31	0.99	1246

Table 2: Metrics for xml generation from user prompts using custom trained model

Prompt #	Jaccard Similarity		Cosine Similarity	Levenshtein Distance
	Tags	Data		
1	1	1	1	0
2	1	1	1	0
3	0.66	0.81	0.99	38
4	1	1	1	3
5	1	1	1	0

Observation

In the analysis of Prompt #1, both OpenAI and the custom model effectively handled the intentional typographical error. However, with Prompts #2, #3, and #4, OpenAI introduced extraneous and irrelevant sections from the template XML, which negatively impacted both the Jaccard similarity and Levenshtein distance metrics as shown in Table 1 & 2. Despite this, the key information necessary for constructing the simulation in APS was successfully extracted. In contrast, the SpaCy model, which was trained on specific data, did not exhibit this issue and generated a more accurate XML output. For Prompt #5, OpenAI unexpectedly converted units of measure, while the SpaCy model did not. These findings highlight the significant role of sentence structure in the conversion of prompts into structured XML.

Case study 2

Intent Troubleshoot a built simulation in APS shown in Figure 4, using user prompts in a sentence structure with intentional grammatical errors and typos.

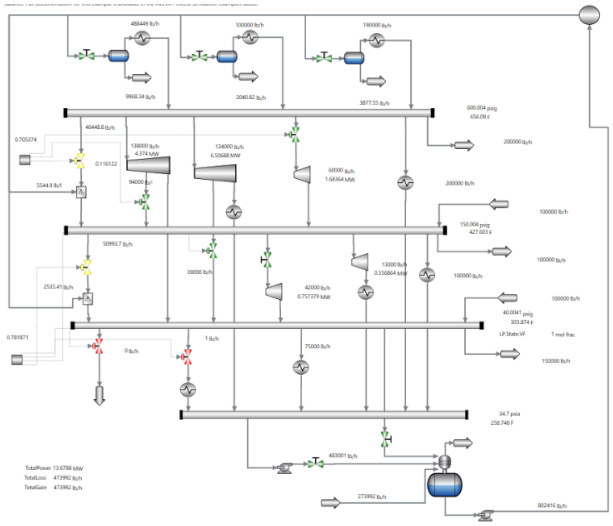

Figure 4. Refinery Steam Balance simulation built in APS.

User Prompts

Prompt 1: Find how many sources, sinks, and valves are present in the simulation

Prompt 2: Find the count the number of sinks, valves and sources1 in the simulation

Prompt 3: Find all Source where temperature is greater than 200 K

Prompt 4: Find all Valves with DP more than 25 kPa

Prompt 5: Can you tell which valves have a DP is greater than 33 kPa

Table 3: Metrics for xml generation from user prompts using pre-trained ChatGPT model

Prompt #	Jaccard Similarity		Cosine Similarity	Levenshtein Distance
	Tags	Data		
1	1	1	1	1
2	1	1	0.95	48
3	1	1	1	1
4	1	1	1	1
5	1	0.7	0.99	236

Table 4: Metrics for xml generation from user prompts using custom trained model

Prompt #	Jaccard Similarity		Cosine Similarity	Levenshtein Distance
	Tags	Data		
1	1	1	1	0
2	1	1	1	0
3	0.05	0.05	0.89	283
4	0.5	0.5	0.92	172
5	0.5	0.5	0.87	282

Observation

The OpenAI model successfully extracts the correct information from the APS simulation for prompts #1, #3, and #4, as indicated by the metrics in Table 3. However, for prompt #2, the XML output contains irrelevant information, which negatively impacts the cosine similarity and Levenshtein distance scores as shown in Table 3 & 4. In the case of prompt #5, the OpenAI model generated incorrect tags, specifically "valve1" and "valve2," along with duplicate information in an improper XML format. This error adversely affected the cosine similarity and Levenshtein distance values, ultimately leading to a failure to return a valid response to the user prompt.

The SpaCy model successfully parsed and generated XML for Prompts #1 and #2 when no conditions were included. However, when conditions were introduced, the model failed to generate the relevant sections and was unable to identify the items for Prompts #3 and #5. While the item for Prompt 4 was correctly identified, the associated condition was not parsed accurately. Overall, the SpaCy model performed better with smaller templates but struggled to handle the added complexity introduced by diverse section types.

Case Study 3

Intent Create a fluid in APS to calculate mixture properties and predict vapor liquid equilibrium using descriptive user prompts

User Prompts

Prompt 1: Get Vapor fraction for a fluid containing Water, Ethanol, Acetone, and molar composition of 0.5, 0.4, 0.1 using system method NRTL and phase vapor and single liquid equilibrium. Show the fluid details along with mixture liquid density property data at 400k and 200 psi and binary interaction data. Also perform flash calculations at 400 K and 200 psi with the given molar composition.

Prompt 2: Get VF for a fluid with Water, Methanol, Ethanol, Acetone, and molar composition of 0.5, 0.2, 0.2, 0.1 using system method NRTL and one phase vapor liquid equilibrium. Show the fluid details along with mixture density property data at 500 K and 1000 kPa and binary interaction data. Also perform flash calculation at 500 K and 1000 kPa with the same defined composition.

Prompt 3: Get VF for a fluid with Water and Acetone, and molar composition 0.6, 0.4 using method NRTL and phase vapoingle and single liquid. Show the fluid details along with mixture liquid density and binary interaction data. Perform flash at 450 K and 1000 kPa

Prompt 4: Get VF for a fluid with components Methane, Ethane, Water, Butane, and molar composition 0.2, 0.2, 0.4, 0.2 using method SRK and system phase vapor liquid equilibrium.

Prompt 5: Get VF for a fluid with components Methane, Ethane, Water, Butane, and molar composition 0.2, 0.4, 0.2, 0.2 using system method SRK and phase vapor single liquid equilibrium

The information extracted from the prompts are categorized as follows,

1. Component Names
2. System method and Phase
3. Fluid properties – details, mixture properties, interaction parameters
4. Flash conditions – Temperature and pressure (along with unit of measure)

Table 5 presents the information extracted by both the OpenAI and custom models, with "Y" indicating correctly extracted information and "N" representing either incorrect or missing information in the XML format.

Table 5: Status of information extracted from user prompts using Custom and OpenAI model

Category		1	2	3	4
Prompt 1	OpenAI	Y	Y	Y	Y
	Custom	Y	Y	Y	Y
Prompt 2	OpenAI	Y	N	Y	Y
	Custom	Y	Y	Y	Y
Prompt 3	OpenAI	Y	Y	Y	Y
	Custom	Y	N	Y	Y
Prompt 4	OpenAI	Y	Y	Y	Y
	Custom	Y	Y	N	Y
Prompt 5	OpenAI	Y	Y	N	Y
	Custom	N	N	N	N

Table 6: Metrics for xml generation from user prompts using pre-trained ChatGPT model

Prompt#	Jaccard Similarity		Cosine Similarity	Levenshtein Distance
	Tags	Data		
1	1	0.89	0.99	17
2	1	0.89	1	1
3	1	0.85	0.99	27
4	1	0.7	1	7
5	1	0.79	1	6

Table 7: Metrics for xml generation from user prompts using custom trained model

Prompt#	Jaccard Similarity		Cosine Similarity	Levenshtein Distance
	Tags	Data		
1	1	1	0.99	28
2	1	1	1	2
3	1	1	0.99	269
4	1	1	0.99	21
5	0.05	0.05	0.78	948

Observation

OpenAI successfully extracted the information from prompt #1 and generated the XML with the expected unit of measure conversion, matching the XML template. However, for prompts #2, #3, and #4, the phase was extracted as VLLE instead of VLE, which could lead to incorrect results in APS depending on the fluid behavior. Additionally, for prompts #4 and #5, fluid properties were included in the XML and APS results, despite not being requested by the user. It is important to note that while the component names, system method, and flash conditions are correctly extracted from the user prompt, the

phase is inaccurately determined due to the sentence structuring of the prompt.

The custom model successfully predicted the outputs for Prompts #1, #2, and #3. However, Prompt #4 yielded an incorrect result due to an error in parsing the MixturePropertyData tag, which led to a failure during XML parsing. In contrast, Prompt #5 resulted in a complete failure in XML creation, with entities being misclassified as shown in Table 5. Notably, the prompts in question are largely like Prompt #4, differing primarily in sentence structuring and a typo error.

CONCLUSION

The conversion of user prompts into a standardized XML format provides significant flexibility for modifying underlying APIs, facilitating easier updates to the simulator. This process enables the rapid development of new XML-to-API generators compatible with various simulation tools. Both the custom and pre-trained models can address a variety of challenges when extracting information from prompts, including grammatical errors, typos, repetitive unit operations, abbreviations, and unit of measure conversions. These models successfully convert the extracted data into a structured XML format, which can subsequently be used to generate simulations in APS. However, the quality of the XML output is influenced by factors such as the sentence structure of the prompt, the specificity of model variables, and the units of measure employed. Based on the case studies, it can be observed that the custom model excelled at standard queries, while the OpenAI model handled complex prompts with intricate sentence structures more effectively, despite occasionally introducing unexpected XML changes.

The pretrained GPT-3.5-turbo OpenAI model is readily accessible and easy to use, whereas the SpaCy model, being custom-trained, requires significant investment in terms of labor, cost, and ongoing maintenance as the model needs to be retrained with new datasets.

DIGITAL SUPPLEMENTARY MATERIAL

The supplementary material can be found with the LAPSE ID: **LAPSE:2025.0019**

REFERENCES

1. Julien B, Cal D. The role of process engineering in the digital transformation. *Computers & Chemical Engineering* (2021)
2. Zeinab H, Shokoufe T, Mohammad HEA. Application of AI in Chemical Engineering. *InTechOpen* (2018)
3. Venkat V. The promise of artificial intelligence in chemical engineering: Is it here, finally?. *AIChE Journal* 65:466-478 (2018)
4. Konstantinos S, Charis N, Paris V, Elias K, Patrik K, Serafeim M. Enhancing property prediction and process optimization in building materials through machine learning: A review. *Computational Materials Science* 220 (2023)
5. Mohammad A, Bassel S, Mohamed M, Enas S, Maryam A, Abrar I, Muaz R, Abdul O. Progress of artificial neural networks applications in hydrogen production. *Chemical Engineering Research and Design* 182:66-86 (2022)
6. Haley H, Mohannad N, Xinyan H, John G. The role of large language models (AI chatbots) in fire engineering: An examination of technical questions against domain knowledge. *Natural Hazards Research* (2024)
7. Chung L. What Is the Impact of ChatGPT on Education? A Rapid Review of the Literature. *Education Sciences* 410 (2023)
8. Mohannad N. Machine Learning for Civil and Environmental Engineers: A Practical Approach to Data-Driven Analysis, Explainability, and Causality. New Jersey: Wiley (2023)
9. Alipour H, Nick P, Kohinoor R. ChatGPT Alternative Solutions: Large Language Models Survey. arXiv preprint arXiv (2024)
10. Dhiah S. Jaccard Coefficients based Clustering of XML Web Messages for Network Traffic Aggregation. *Journal of Al-Qadisiyah for computer science and mathematic* 11:82-91 (2019)
11. Faisal R, Teruaki K, Masayoshi A. Semantic cosine similarity. 7th international student conference on advanced science and technology ICAST 4 (2012)
12. Seo N, Sangwoo K, Cheonyoung J. A Lightweight Program Similarity Detection Model using XML and Levenshtein Distance. FECS 3-9 (2006)