Auto-manual: An LLM-integrated Assistant for the Human-machine Interface Troubleshooting

Haozhou Zhou
Dec 12, 2023
CS6140 Final Project

1. Introduction

Manufacturing equipment is getting complex. It has not remained purely mechanical as it was a few decades ago. Informatics systems, along with the Internet of Things (IoT), cloud computing, and machine learning, are integrated with manufacturing equipment to create highly productive and interconnected manufacturing systems. Only with the aid of these newly updated systems are the manufacturing industries able to satisfy the currently vastly expanding and highly customized markets.

What, then, would be the cost? As these systems have shifted to an interdisciplinary complex, operational errors, faults, and malfunctions have become ubiquitous. The required knowledge and expertise for troubleshooting are increasing accordingly. Siemens Co., Ltd. has been striving for solutions to these challenges. They introduced standardized human-machine interfaces (HMIs) as versatile control panels that can be mounted on general manufacturing equipment. For those that have been equipped with Siemens' HMIs, engineers can use them to check the operational status and take proper actions against latent or existing malfunctions and errors. However, it is not a permanent solution. During manufacturing processes, malfunctions may simultaneously involve the execution system, logic control system, or radio frequency identification (RFID) system. As such, they could be too complicated to be resolved by one person's effort. What most likely happens is a long-period, multi-expert participated diagnosis, guidance, and repair. And the cost, in many ways, is massive and unacceptable for small and medium-sized businesses.

Therefore, we're seeking a quicker, more affordable solution without sacrificing any effectiveness. The large language models (LLMs), represented by OpenAI’s ChatGPT, are arguably the promising

tool to achieve this goal. There are three arguments about why LLMs can fulfill this job and how they are going to approach the challenge:

Figure 1 Siemens Human-machine Interface Products Lineup

(1) **Troubleshooting during manufacturing processes needs interdisciplinary expertise, and LLMs are featured by interdisciplinary knowledge acquisition and representation.** LLMs gain the capability from comprehensive training on a wide-ranging dataset, the use of advanced machine learning techniques, architectural design, and ongoing refinement and updating processes. These factors collectively contribute to the model's ability to understand, process, and generate information across a multitude of disciplines.

(2) **Time is of the essence. LLMs could offer instant guidance or suggestions for troubleshooting, which might greatly mitigate time losses on communication and waiting that the old method costs tremendously.** LLMs are pre-trained on large datasets before deployment. This pre-training means that the model has already learned a vast amount of information and patterns in language, allowing it to quickly generate responses based on this pre-existing knowledge without needing to learn from scratch for each query.

Their responding speed could also benefit from its transformer architecture, particularly as we learned from the CS6140 lecture, of which the attention mechanism and layer normalization can quickly process and generate information.

(3) **There are ample open resources of application program interfaces (APIs) to engineer LLMs. They make the development of LLM applications much more feasible than ever before.** APIs serve as intermediaries that allow developers to access the functionalities of LLMs without needing to understand underlying mechanisms. For example, the OpenAI API provides access to models such as GPT-3.5 and DALL-E, which developers can use for various purposes ranging from natural language processing tasks to image generation. The open-source frameworks we learned from the CS6140 course: Hugging Face's Transformers library, provide pre-trained models and pre-built APIs. These resources make it easier for developers to implement LLMs without starting from scratch. Specifically, the Transformers library offers thousands of pre-trained models in over a hundred languages, which can be used for tasks across text classification, translation, summarization, and so much more.

2. Method and Experiment

To validate this primitive idea, a framework named "Auto-Manual", consisting of Optical Character Recognition (OCR), GPT-3.5-turbo, and speech synthesis was proposed. The framework can extract fault messages from human-machine interfaces, use GPT-3.5-turbo to offer maintenance advice, and then employ a voice model to deliver this advice, automating entire troubleshooting workflows with an add-on of artificial intelligence. Engineers are required only to upload a picture

of the HMI with error messages, after which they will receive guidance from Auto-Manual within minutes.

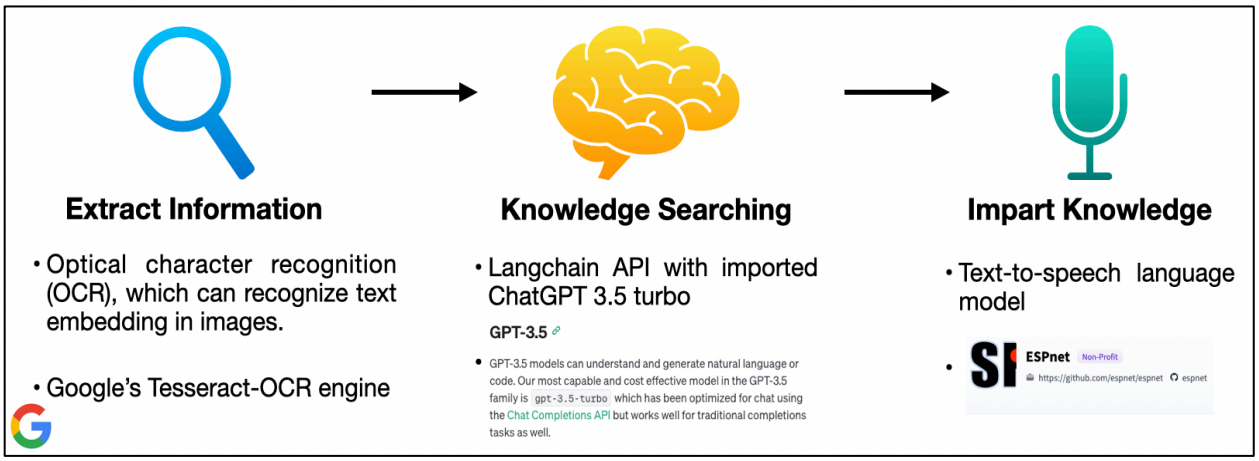


Figure 2 the "Auto-Manual" Framework

The experimental environment was set up in a conda virtual environment with Python 3.9 as the interpreter. The most classic and widely adopted OCR engine in Python, pytesseract, was utilized in the first stage of Auto-Manual for information extraction. This engine incorporates a combination of machine learning and computer vision algorithms, enabling it to process the image line by line, and then segment lines into words and words into characters. Thereafter, spell-checking and correction based on context would be applied to the recognized texts as the finalized output. With the extracted information as input, GPT-3.5-turbo will analyze it and generate instant feedback. This process was streamlined by LangChain API, an open-source framework for developing applications powered by LLMs. It features a function that is built-in "context-aware", which allows Auto-Manual to learn from a brief prompt that tunes GPT-3.5-turbo with a characterization as an "Industry 4.0 expert and sophisticated industrial engineer". In the final stage of Auto-Manual, an end-to-end speech processing toolkit, ESPnet, was imported from the Hugging Face API to read out GPT's answer aloud. Only the text-to-speech module from ESPnet was leveraged. This module's architecture consists of a pre-trained transformer and multiple NLP machine learning models that can generate 200-word speeches within seconds. To better showcase Auto-Manual, the streamlit API was leveraged to wrap the entire framework in the form of a web application. Streamlit, which requires zero front-end development experience,

can transform Python scripts into web apps. It works by simply inserting streamlit commands into the scripts. Detailed dependencies are listed in Table 1.

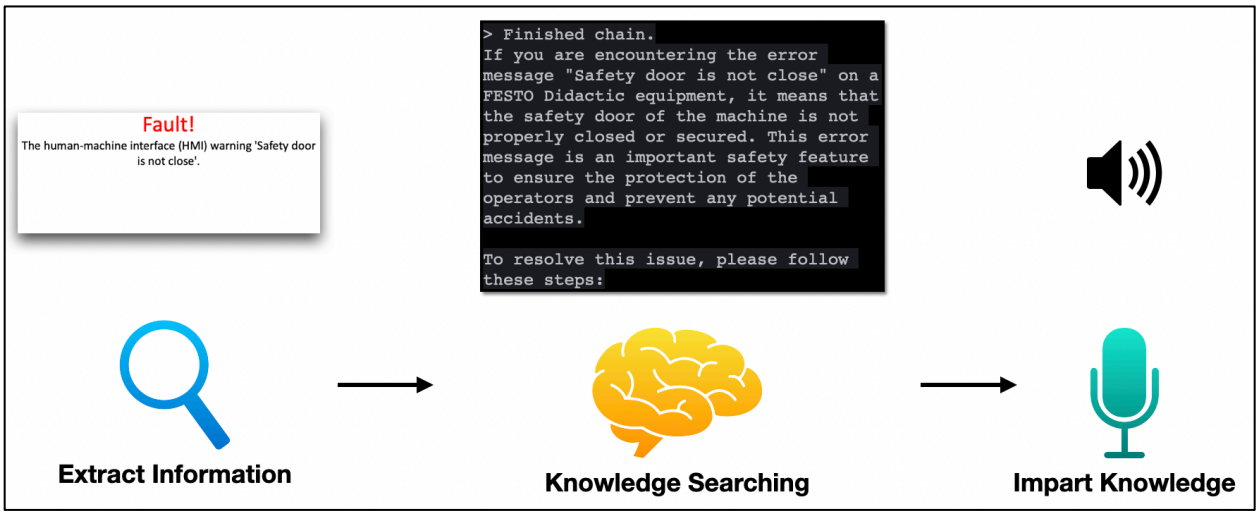


Figure 3 Experiment and Demo of Auto-Manual

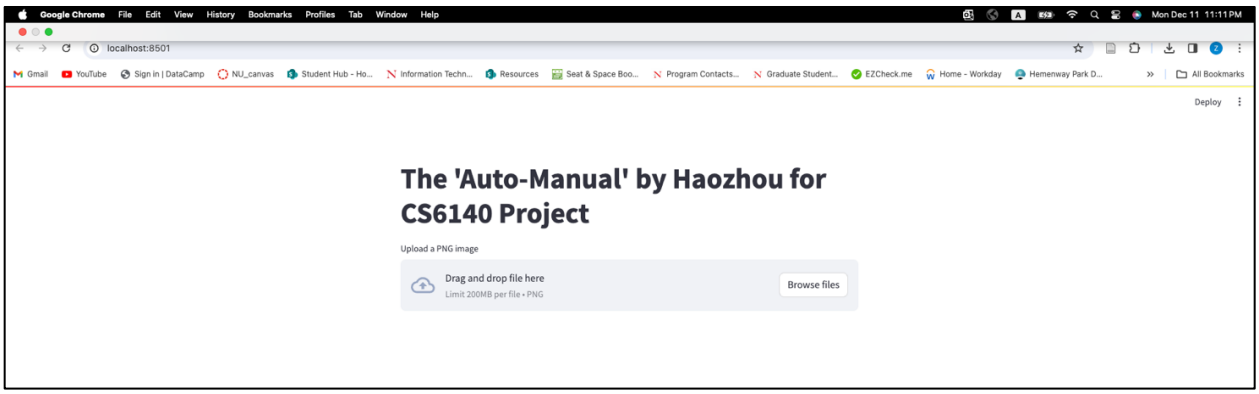
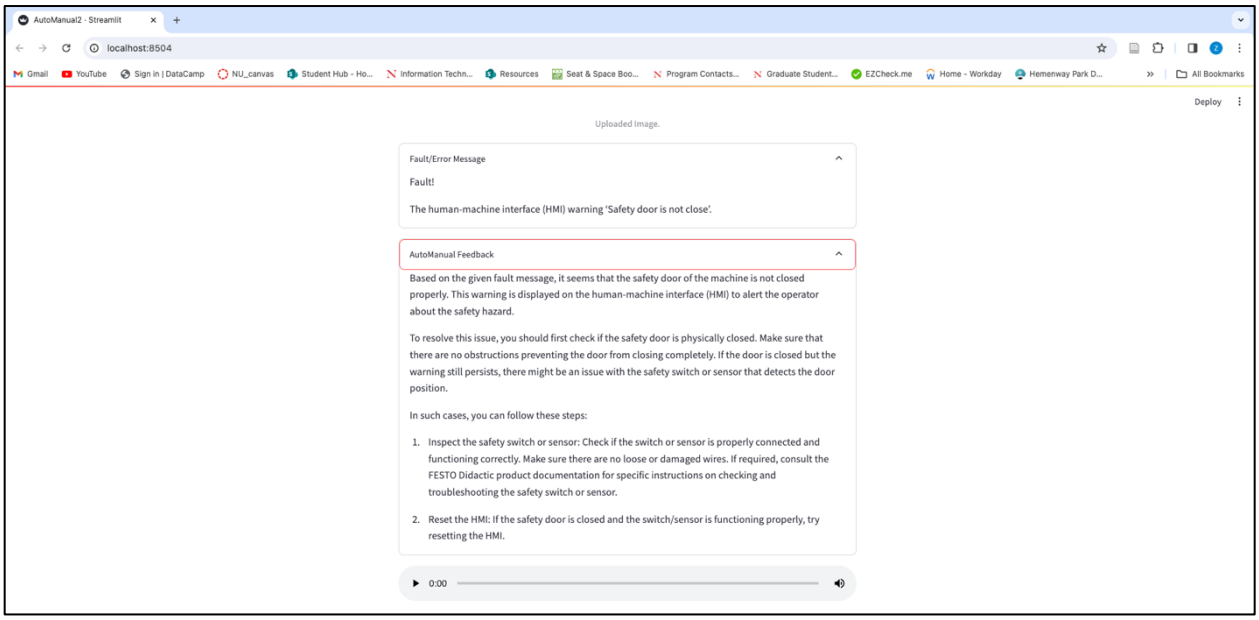


Figure 4 Auto-Manual Demo

For the experiment and demonstration, a simulated HMI error message, “safety door is not closed”, was used. The demo of Auto-Manual incorporates three functions summarized in Table 2. The OpenAI and Hugging Face API keys need to be properly typed in as two constant variables. Then the script can be executed in the Python terminal using the command “streamlit run AutoManual.py”, which will direct it to the default browser to access the web application of Auto-Manual. After successfully executing the scripts, the “Browse files” button can be used by users

to select the simulated image to proceed. Within seconds, both the extracted information and answers generated from GPT-3.5-turbo will be listed on the tab. The answers' voicing will have an audio player interface in which users can decide to either read the answers or play them.



Furtherly Figure 5 Auto-Manual Demo

Table 1 Auto-Manual's Framework Dependencies

Python Libraries	Version
python	3.9
pytesseract	0.3.13
openai	0.28.1
langchain	0.0.248
streamlit	1.29.0
requests	2.27.1
typing-extensions	4.5.0

Table 2 Python Functions of Auto-Manual Framework

Functions	Descriptions	Parameters	Returns
extract_text_from_image	Extracts text from the given image URL using OCR.	image_url (str): The URL of the image.	str: The extracted text from the image.
generate_gpt_feedback	Generates feedback for a given fault information using GPT3.5-turbo.	fault_info (str): The fault information to generate feedback on.	str: The generated feedback from GPT.
synthesize_speech	Synthesizes speech from the given text feedback using Hugging Face's API.	feedback (str): The text feedback to be converted into speech. api_key (str): The API key for Hugging Face.	None: Writes the audio to a file under the same directory with the name of 'audio.flac'.
display_uploaded_image	Displays the uploaded image in the Streamlit app.	uploaded_image: The uploaded image file.	None: Displays the image in the app.
process_and_display_feedback	Processes the uploaded image to extract fault information, generate feedback, and synthesize speech.	uploaded_image: The uploaded image file.	None: Displays the extracted information and feedback in the app.

3. Reflection

The primitive experiment and demonstration for Auto-Manual is a limited and idea-validation procedure. The core of Auto-Manual, which are large language models, needs comprehensive refinement to improve its intelligence in the knowledge of manufacturing industries. To make this happen, manufacturing knowledge embedding needs to be applied to GPT-3.5-turbo. It would also be imperative to interview practical industrial engineers and workers to gather answers about what kind of help they might need most from LLMs. Thereafter, data collection for fine-tuning is necessary, after which the fine-tuned LLM can optimize the quality of generated information and fulfill these needs. If the Auto-Manual develops to become a built-in module of HMIs, these next-generation HMIs might not only be a control panel and notification center; it will be capable of 'thinking', offering actionable suggestions, and still keep the human in the loop, who will use their judgement to take the final action.