Abstract

*In any manufacturing company, there are operators that will need access to lots of information, one of those will come from machine datasheets to ensure efficient operations. The primary issue is the inefficiency in manually searching these documents along with the complexity to retain all that information by a human brain. This project's main goal initially was to develop a chatbot in python using a Generative Artificial Intelligence (Gen AI) integrated with a document intelligence framework to answer operator’s questions efficiently. Unforeseen circumstances led to a change in approach from using a LLM to training a classifier based in a pseudo-chatbot environment. The expected outcome is a chatbot that accurately answers questions (simulating operators) based on these documents, avoiding as much as possible any type of hallucination. The results will be evaluated both qualitatively and quantitatively. Qualitatively, the chatbot’s responses will be assessed for accuracy on the test set. Quantitatively, images of example test results. This solution leverages AI to streamline operations, could potentially reduce mill downtime, and enhance productivity in a critical industrial setting.*

1. **Introduction**

In many industrial workplaces, operators frequently need quick access to detailed information from machine datasheets to ensure smooth and efficient operations. This leads to a problem in the steel mill industry, where the current process requires the operator to search through extensive documentation to find relevant information. This project aimed to develop a chatbot using a Gen AI tool (GPT) integrated with a document intelligence library to answer operators’ questions about these documents in a timely manner. This solution would have leveraged advanced AI to streamline operations, reduce downtime, and enhance productivity in a critical industrial setting. As explained later, the team had to go in a different direction because of unforeseen circumstances. Instead of using a LLM or GenAI tool married with a document intelligence library to generate labels that could be later used to finetune our results and model, we opted to use a pseudo-chatbot environment. Real chatbots are commonly associated with large LLM models such as ChatGPT and Gemini. LLMs are already pre-trained on large datasets that enable them to answer questions on a variety of subjects. Our pseudo-chatbot environment is merely a mechanism that enables us to answer questions that have pre-defined (meaning that the responses that our model generates are not random) answers coming from an external file. The model will be able to accurately classify incoming questions to associated tags. Each tag has an associated answer that we have pre-defined in our evaluation documents.

For steel mill operators, manually accessing information from machine datasheets requires significant time and effort. Ideally, a solution must not only be efficient and reliable for high-pressure situations, but also must be able to properly parse the context and specifications of given requests.

Our proposed solution dataset will consist of various machine datasheets and operational manuals. The expected result is a chatbot capable of accurately answering operators’ questions based on these documents. The initial proposed method involved using a Gen AI model like GPT, combined with a document intelligence library for parsing and understanding the datasheets. Existing implementations of GPT would have been utilized, with modifications to tailor the responses to the specific context of steel mill operations.

The data will include existing machine datasheets and operational manuals from online resources. Ideally, if newer data needed to be integrated, it would be collected through direct input from the operators and maintenance logs. This data would be digitized and formatted to be compatible with a document intelligence library.

The resulting answers from the chatbot would be evaluated in a variety of ways. Qualitatively, the chatbot’s responses would ideally be assessed for accuracy and relevance through operator feedback. Quantitatively, performance metrics such as response time, accuracy rate, and user satisfaction scores could have all been used. Statistical tests could have been conducted to compare the chatbot’s performance against manual search methods. Because of time constraints and unexpected roadblocks that the team encountered, the team ended up having to pivot away from using an LLM. Instead of having metrics to measure response rate, and user satisfaction rate, and methods to perform statistical tests, only test accuracy is being measured.

1. **Background/Related Work**

The implementation of a document-intelligent chatbot for steel mill operations increases the advancements in Gen AI to address the issues in the process of searching documents manually. Project approach is supported by the findings of [1] that explored a few impacts of this technology AI for students. Their study was done by applying a survey with students and analyzing the potential of Gen AI to significantly reduce the cognitive load on users, a benefit that is similar and could be applicable to industrial environments where they’re in a constant learning phase and they need to quickly access and interpret machine datasheets. Similarly, [2] did an analysis of ChatGPT’s capabilities, demonstrating how conversational AI can transform user interactions by providing accurate and contextually relevant responses. This also aligns with the project’s initial goal of developing an intelligent chatbot that will prevent hallucinations, minimizing it and enhancing the operational efficiency.

The review by [3] was applied to the impact of Gen AI in industrial applications, like the main goal of this project, by underscoring the transformative potential of AI technologies in improving productivity and reducing downtime. By referencing these studies, the initial project goal was to build on existing papers to create a chatbot solution for steel mills or similar manufacturing companies. The predicted output could have included improving response times to make operators' jobs easier and faster.

BERTbase, a pretrained implementation of the BERT model that aims to answer reading comprehension questions, can be applied to different downstream tasks that are either a single sentence, such as sentiment analysis tasks, or pairs of sentences, such as question answering tasks with a given context [4]. It has a total of 110M parameters, 12 layers or transformer blocks, a hidden size of 768, and 12 total attention heads. BERTlarge has 340M parameters in total, 24 layers or transformer blocks, a hidden size of 1024, and 16 total attention heads. Since BERT’s transformer uses bi-directional self-attention while GPT’s uses constrained-to-left-context self-attention, BERTbase could not be made the same size as OpenAI GPT as intended. The BERTbase model has been tested on the SQuAD 2.0 dataset and evaluated using the F1 score and EM score. Four different experiments are performed using the same preprocessing, pre-trained BERTbase model, and training and validation dataset, but differ in the training arguments. When the Training Epochs parameter is increased from 3 to 6, the EM and F1 scores decreased and the model overfitted. When the Learning Rate parameter was set to be anything other than 2e-5, the EM and F1 scores were both defected. Each of the experiments uses three parameters: training epochs, weight decay, and learning rate. The preprocessing was performed by ignoring questions that were unanswerable, removing the extra spaces, splitting the question and context into sets of tokens that also contain the special token [CLS] at the beginning and [SEP] token between both the question and context and after the last token of the context, and splitting the exceeded length of 300 token question-context pair sample into many samples containing that question paired with the remaining part of the context. Out of the four experiments, the highest EM score is 61.12 and the highest achieved F1 score is 72.5. The authors concluded that the unanswerable questions problem looms large and that using a larger model with more pre-trained parameters such as BERTLarge would be a great starting point and evaluated in the QA domain through one-shot and zero-shot learning.

MedQA, a reading comprehension task introduced by [5] to answer questions in clinical medicine using knowledge from a large-scale document collection, is handled by using a novel modular end-to-end reading comprehension model based on LSTM networks and dual-path attention architecture called SeaReader. SeaReader attempts to both leverage information in large-scale text and use end-to-end training on weak labels to address the challenges that the MedQA task poses for language understanding, especially when compared with existing reading comprehension datasets. Major challenges of the MedQA task include lack of professional knowledge, the diversity of questions, determining the best answer, and interpreting large scale text. Because of the sophistication that the medical field introduces, questions related to clinical medicine generally require the expertise of someone in the field to answer them adequately. This is true as opposed to other reading comprehension tasks where the minimum requirements for question answering is language proficiency and commonsense reasoning. For example, it might be necessary to acquire operator feedback to assist in evaluating the quality of our chatbot’s responses when questions are answered on our training dataset of operating manuals. For some questions to be adequately answered, external information may need to be gathered for the chatbot to provide more precise responses. Because of how broad the medical field is, the model proposed in [5] should be able to answer questions on a wide range of topics. The model should be able to go from answering questions regarding making clinical diagnoses based on a description of a patient’s condition to suggesting the best course of treatment. Ideally, the chatbot model that the team is proposing should be able to answer questions about operator manuals that are associated with steel mill machinery. However, if the availability of steel mill operator manuals online is not sufficient, the addition of non-steel mill specific technical datasheets and operator manuals to the training dataset should be considered. When the model is interpreting large-scale text, it is pertinent that the model the team uses can effectively synthesize precise answers based upon the context of the questions asked, and the documents involved. It is also important that the steel mill operators who use this bot in the future can trust the output of its responses. In other words, the model should be able to pinpoint the line or lines used to answer questions that pertain to one or more operator manuals. When the model answers a question, the model should output the operator manual or datasheet that it believes the question pertains to, the line(s) used to answer the question and generate text indicating the comfort level of a response provided based on a confidence score calculated based on available evaluation metrics. That additional feedback on top of the response provided will also assist in debugging for developers. If the chatbot is consistently indicating that poor quality responses are being given based on available evaluation metrics, the developer will know to take additional approaches related to pre-training and fine tuning.

1. **Approach**

The project’s approach is described as the following: First, pdfs on various topics of Steel Mill practices (health and safety, acceptable ore quality, installation instructions) with an emphasis on machine operation manuals are gathered to a trainable dataset. These PDFs are then labeled utilizing tags based on the given topic or instruction type. This dataset is then fed into a model for training purposes, where we plan to utilize a grading system for tuning based on the accuracy, length, and overall structure of answers. This entire process can be performed within the confines of an online data system, making it efficient for tuning.

The approach used in the project was followed by the sequence detailed in Figure 1, which was determined by first, the documents storage that was gathered during the preprocessing phase of the project.

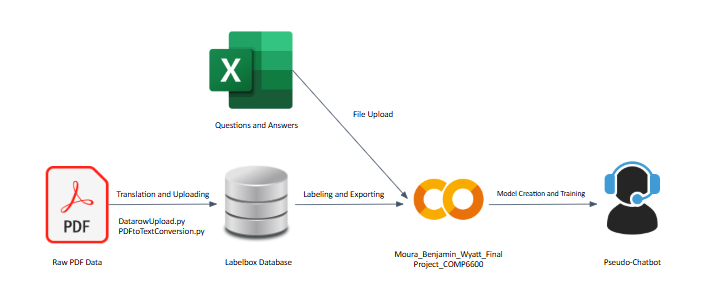


Figure 1- Flowchart of the Document-Intelligent Chatbot

Additionally, the Intent classifier will be trained in our pseudo chatbot environment, and the whole process is hosted in Google Colab utilizing the python notebook format. Each intent refers to a specific document that exists in our dataset. The dataset included 18 machine datasheets and operational manuals. Two of them had to be thrown away because of formatting mistakes that were induced by the limitations of the pdf-to-text converter that we used. Ironically, the data that we labeled in LabelBox ended-up not being used due to a forced change of approach. The model is trained on 16 documents. Each intent has a number of questions and associated tags. The classifier is supervised because each question in our training set has an associated tag. Another notable change from the initial proposition of the project was the utilization of answers that fall within certain categories over the usage of a large language model. This was based on the small amount of data we had gathered being significantly less than the amount utilized for these chatbots, so manually defining responses was in turn less strenuous.

1. **Code Development**
   1. Code Explanation

The chatbot’s code utilizes a number of different functions in order to properly recognize and classify intent. For the labeling process, two scripts were utilized in order to process and upload a selection of PDF documents, being DatarowUpload.py and PDFtoTextConversion.py. PDFs containing data related to Steel Mill machines, processes and regulations are converted into a text format, stripped of any untranslatable bytes or other unusable data, and saved into a plain text format. Once this has been done, the data is uploaded utilizing Labelbox’s python module to cleanly provide the data in its own dataset. The data is then labeled using Labelbox; the exact process of doing so will be further covered in Section 5.

Once uploaded and labeled, the data is imported to a Google Colab python notebook. Here, both the labeled data for intents and associated questions are imported, cleaned, and synced so that questions and intents can be related in dictionaries. A variable to track the total numbers of words found in both the labeled data and questions is created (known as a “bag of words”), and acts as the main tool for transforming sentences into binary lists, the data type necessary for providing the inputs and outputs. Next, the questions and related data to intents is translated into the bag of words format, which is then stored into variables for usage as Neural Network training and testing data. Utilizing tflearn, a library composed of systems for creating and utilizing Deep Learning Neural Networks, a DNN is configured and trained on the data described above. Finally, the user is provided with a place where they can input their own custom questions, which the deep learning model interprets and provides a relevant answer. For efficient testing, a command for running all of the created test questions was also provided.

* 1. Code Reuse

Labelbox’s documentation and guides [6] were integral in our creation of the chatbot and were used throughout the process. However, additional components and tools were necessary to develop on our own, including code for importing and cleaning our data from both Labelbox and our own questions as well as the interface for communication for the user. All of these references have been noted when relevant.

1. **Experiment**

The primary experiment for this project is to develop and test a chatbot designed to assist operators in a steel mill by answering questions based on machine datasheets and operational manuals. For that, 18 sample documents related to Steel Mill and Machine Operations were gathered from the internet. Figure 2 can illustrate well one example of a technical datasheet serving as Machine Operation and Figure 3 shows an example of a Steel Mill document:





Below is the document list used as a dataset for training the chatbot. Due to the limitations of the pdf to text converter that we used, two of the operator manuals had to be discarded.

* Steel Mills Health and Safety.pdf
* Steel Mills Health and Safety.pdf
* Shear Break and Roll Machines.pdf
* Scrap Iron and Steel Manual.pdf
* Rolling-Mill-Instructions-New-Durston.pdf
* Rolling Mills Overview Techna.pdf
* Pollution Control.pdf
* IPSi2400X-Steel-Mill-MDS-Manual.pdf
* gms-2500l-ss-vertical-washing-machine-data-sheet.pdf
* gms-2500c-horizontal-washing-machine-data-sheet.pdf
* gms-1600l-vertical-washing-machine-data-sheet.pdf
* gms-1600l-ss-vertical-washing-machine-data-sheet.pdf
* gms-1600-heated-roller-press-data-sheet.pdf
* gms-1200c-horizontal-washing-machine-data-sheet.pdf
* gms-2k-pneumatic-two-component-sealant-pump-data-sheet.pdf
* collapsible-support-stand-data-sheet.pdf
* automatic-single-head-upcut-mitre-saw-data-sheet.pdf
* automatic-end-milling-machine-data-sheet.pdf

To evaluate the chatbot’s performance, a metric table was created with some questions to test the chatbot and evaluate if its response matched the predicted intent of our question. These metrics were collected by training and questioning the model, and the results were calculated by taking the accuracy of the predicted tags versus the team’s manually assigned predictions. The team members initially wanted to simulate real-world scenarios by asking the chatbot questions and providing ratings based on the helpfulness and correctness of the answers. This feedback could have been crucial in identifying areas for improvement and ensuring the chatbot met the practical needs of its users. Because of a change in approach, providing a rating for each predicted answer that the model outputted became untenable. To ensure that the intent classifier would work with our dataset, we created one general answer that attempts to satisfy the task of answering all questions related to a specific document. To be clear, each document has their own unique answer. For example, if the model is asked any questions about the Collapsible Support Stand, one answer was curated in the form of a summary that will be sufficient to answer any questions related to its description, key features, and relevant technical details. If the team opted to rate each answer on a scale from 1-5, most of them would either be a 1 if the predicted response didn’t match the actual response or a 5 if the predicted and actual response both matched. The rating metric becomes useless and uninteresting. Thus, instead of using any sort of rating system, test accuracy was utilized instead. The test accuracy is measured by feeding our test set into our pseudo chatbot environment and then recording the number of times that the tag predicted matched the actual tag for that specific test question. The detailed results of our experiments are presented in the evaluation table below.

Each of the question evaluations utilized the following format :

User: What is the 1600 Heated Roller Press?

Bot: The GMS 1600 Heated Roller Press has the following key features...

Figure 4- Example of Pseudo-Chatbot Response

These evaluations were recorded, and any misclassifications were noted at the end of the testing period. This lead to the following results:

Accuracy of Pre-Tuned Model: 0.8235

Accuracy of Post-Tuned Model: 1.0

Within the predefined set of testing questions, an accuracy of 1.0 was achieved after additional tuning (To clarify: as the project has shifted to a pseudo-chatbot, tuning indicated the addition of more examples and information in certain intents). Regardless, both the pre- and post-tuning models displayed excellent competency in identifying each intent, with both accuracies being high. One noteworthy test was the ability to differentiate between various models of the same machine (Ex: “1600l” and “1600l SS”), indicating that the Deep Learning model had properly identified certain key features (such as “SS”) in order to distinguish between similar questions.

An overview of each question from both training and testing is available in the attached “Evaluation\_Table” file under “Training” and “Testing” respectively.

1. **Division of labor**

As pre-defined in topic 1 and 2, the main goal for the project is an alternative to turn the inefficient manual searching into an effective way with the adventure of Gen AI. The idea originated from Vitor Moura that is an Engineer that works in a Steel Mill manufacturing company, so he was responsible mainly to write the project proposal and share with colleagues that joined the team, Daniel was the first to join and he was responsible for adding the proposal text to proper layout and Connor joined later bringing suggestions and adding more content to the proposal.

After the proposal, the team had a first meeting using the "Discord" channel (a dedicated space in Discord website [7] where users can communicate with each other) and it was decided properly the next steps and division of labor. The division of labor for this project was clearly defined to ensure a better collaboration and further completion.

The team held weekly meetings via a Discord channel to coordinate their efforts. Connor and Vitor were responsible for gathering 18 documents to train the chatbot. Connor initiated the setup with Labelbox by translating and uploading documents while Vitor and Daniel focused on the midterm report. The labeling process through Labelbox was divided equally among the team members. Vitor took the lead in writing the final report, Connor managed the configuration and setup of the chatbot, and Daniel made the code adjustments and modified the evaluation table. Daniel and Connor also focussed on drafting questions, answers, and intents in both the pre and fine tuning stages. Daniel also worked on retrofitting the supervised intent classifier to work in a so called ‘pseudo-chatbot’ environment to make it easy to view and extract predictions on the test set made by the model through a simple command into our evaluation table for both pre and post finetuning. Daniel also worked so that the code would extract and print the most recent accuracy to the evaluation table.

To finish, Connor and Daniel were responsible for the final presentation, but Vitor made the slides to support the presentation. The division of labor in the project improved efficiency and collaboration, allowing team members to focus on specific tasks. The group member’s weekly meetings via Discord ensured clear communication and coordination, while the balanced workload helped the team moving forward to the outcome. Leveraging individual expertise, such as Vitor’s industry experience, Connor’s coding skills, and Daniel’s attention to details, resulted in a well final paper and project to the course of COMP6600 - Artificial Intelligence.

1. **Conclusion**

It was enlightening to define and create a chatbot, and a significant amount was learned from the labeling and training process. A few key improvements could be made, however: Most notably, the process of converting PDFs could utilize additional refinement, with other options being investigated but unable to be utilized in the timeframe for the project. Additional labeling could also be done to provide further context for individual topics, and adding certain additional filters such as removing certain commonly used words like “the” or “and” (also known as “stop words) could benefit the training of the chatbot. Despite all of these potential improvements, the results of the pseudo-chatbot were still pleasing to see.

The pseudo-chatbot was able to accurately predict on a range of testing questions, and was able to both closely predict on similar data as well as handle words not seen within its training. One strength this design has over an LLM is that once the system has been loaded locally, neither an internet connection nor significant processing power is required to get information. This could be greatly beneficial for workers in places and conditions that a standard chatbot could not, and provides an excellent stepping stone for future development in helping steel mill workers across the globe.

1. **References**

# Abbas, M., F.A. Jam, and Khan. 2024. "T.I. Is it harmful or helpful? Examining the causes and consequences of generative AI usage among university students."

# Bansal, G., V. Chamola, and A Hussain. 2024. "Transforming Conversations with AI—A Comprehensive Study of ChatGPT." *Cogn Comput* 2487–2510.

# Kar, A.K., P.S. Varsha, and S. Rajan. 2023. "Unraveling the Impact of Generative Artificial Intelligence (GAI) in Industrial Applications: A Review of Scientific and Grey Literature." *Glob J Flex Syst Manag 24* 659–689 .

# Mosaed, A.A. Hindy, H. and Aref, M. “BERT-Based Model for Reading Comprehension Question Answering,” pp.52–57, Nov.2023, doi: <https://doi.org/10.1109/icicis58388.2023.10391167>.

# X. Zhang, J. Wu, Z. He, X. Liu, and Y. Su, “Medical Exam Question Answering with Large-scale Reading Comprehension,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, doi: <https://doi.org/10.1609/aaai.v32i1.11970>.

# Labelbox, "How to train a chatbot," Labelbox, [Online]. Available: https://labelbox.com/guides/how-to-train-a-chatbot/. [Accessed 12 October 2024].

# Discord, " Discord - Group Chat That’s All Fun & Games," [Online]. Available: https:// https://discord.com/. [Accessed 08 October 2024].