Digital Bouncers — Secure Smart Home Solution with Large Language Models

Eric Chen eyc004@ucsd.edu
Phu Dang pndang@ucsd.edu
Daniel Li ddli@ucsd.edu
Bofu Zou bzou@ucsd.edu
Abed El-Husseini aelhusseini@deloitte.com
Nimu Sidhu nimsidhu@deloitte.com

Abstract

As large language models and Generative AI (GenAI) products are becoming increasingly common in people's daily lives both at home and at work, it is important to ensure that these products are behaving like how they are intended to. In general, LLMs and GenAI products are known to processes more efficient, ruling out the need for human input. But with that comes a lot of trust in their abilities to always deliver outputs that are inline with the original purpose. As a result, there is an emerging trend of setting up guardrails to ensure inputs and outputs satisfy some sort of criteria to be deemed safe. However, with the rise of smart homes, we recognize the tangible effects and risks that come with allowing AI to dictate the settings of a home, and the dearth of guardrails for these scenarios. Smart homes collect data and know the house better than its residents do about daily energy consumption and when exactly the lights turn on and off. Therefore, current and prospective smart home residents need a way to understand interact with the smart home systems in a safe and secure manner.

For your Quarter 1 Project, you won't link a website, but will link your code. For your Quarter 2 Proposal, you won't link either of these. For your actual Quarter 2 Project, you'll link both of these.

Website: https://github.com/abc
Code: https://github.com/pndang/digital-bouncers

1 Introduction

1.1 Background

1.1.1 The Opportunity

In "The great acceleration: CIO perspectives on generative AI" by the MIT Tech Review and Databricks, Schaefer—Chief Health Informatics Officer at Kansas City VA Medical Center—stated "trust" as the Key to effectively adopting generative AI. The healthcare industry is a great case study for involving a multitude of high-stake processes and parties, such as the use of machine learning models to predict protein structures, assist drug discovery, and track the progression of outbreaks, to chatbots helping front-line staff by transcribing medical notes and answering patients' questions. The same understanding of such broad applications can be applied to other areas, including a smart home chatbot, where privacy concerns are prominent as users decide who gets to use their data, which data, and for what purposes. As more rooms and appliances generate data in homes, the "democratization" of data control and ownership is poised to gain traction to build "trust" in generative AI. The reverse also applies as developers aim to protect the integrity of AI systems by moderating which user inquiries to accept or reject. Meeting these expectations requires the effective and ethical implementation of conversational AI to protect data and systems integrity, safety, and privacy for the collective benefits of those affected most by generative AI.

1.1.2 Conversational AI Risks and Regulatory Compliance

Many prominent works have been done to document, understand, and disseminate the risks and threats of generative AI, notably is the OWASP Top 10 security risks for large language model (LLM) applications, which includes prompt injection, insecure output handling, sensitive information leakage, excessive agency, to model theft. Knowing these preeminent threats allows developers and users of LLM-based chatbots to navigate development environments, workflows, usage, and system architectures with proactive risk-mitigation techniques to address these concerns. Several legislations worldwide have been adopted to protect data integrity and privacy through "data minimization" best practices for transmission, storage, and API-intensive AI workflows, including the EU's General Data Protection Regulation (GDPR) in Article 5(1)(c), and the California Privacy Rights Act (CPRA) in Cal. Civ. Code § 1798.100 (d). It is critical for current and future developers of AI systems to take leadership and proactive steps in implementing best data-handling practices to inspire a future of safe and secure digital interactions with "digital bouncers".

1.2 Our Solution

Digital Bouncers aims to address these concerns with an LLM-based resource-use monitoring chat application with AI risk-mitigation for physical assets – a trustworthy assistant

users can rely on to ask general questions about smart home energy use. By building upon existing LLM guardrails, our application hopes to demonstrate a feasible solution through guardrails to common problems in LLM applications at the input, intermediary, and output level.

In terms of our actual chat bot application, we incorporated two main features/workflows:

- For questions that can be answered by performing SQL on the dataset, we've implemented a Text2SQL pipeline that utilizes Langchain agents to perform the necessary SQL operations on the data.

- For other questions that cannot be answered using SQL, we call a separate general LLM model that answers the question.

As mentioned earlier, we also focus on implementing guardrails to provide a flawless and secure user experience. Below we give a general overview of the guardrails we used throughout the workflow, as well as the problem they guarded against.

- Input Moderation Guardrail: We have a guardrail at the input level that guards against of topic prompts, as well as harmful or inappropriate prompts.

- Intermediary Guardrail: Since we have two main workflows in the Text2SQL and general LLM; we have an intermediary guardrail that determines which workflow to use according to the question asked.

- RAG Check Guardrail: We have a RAG evaluation guardrail that checks whether or not the answer given makes logical sense. Sort of a sanity check on the output response of our chat bot if you will.

- Output Moderation Guardrail: We will have a guardrail blocks/rewrites responses to conform to our predefined guidelines. This guardrail will also perform a basic syntax check.

In summary, our goal is to provide users with secured recommendations regarding smart home utility bills. We aim to facilitate private, safe, and secure interactions between LLMs and users with robust guardrails and thoughtful interaction flow design that addresses the prevalent risks of conversational AI.

2 Methods

2.1 Data

Below we give a brief overview of the columns in the columns in the data:

2.1.1 Energy Consumption/Generation

- use [kw]: Total power consumption in kW
- gen [kW]: Power generated (likely solar) in kW.
- House overall [kW]: Total household power usage.

2.1.2 Appliance-Specific Consumption

Below is a list of all the recorded energy usage of the appliances/rooms in the smart home.

- Dishwasher [kW]
- Microwave [kW]
- Fridge [kW]
- Furnace 1 [kW]
- Furnace 2 [kW]
- Home office [kW]
- Wine cellar [kW]
- Garage door [kW]
- Kitchen 12 [kW]
- Kitchen 14 [kW]
- Kitchen 38 [kW]
- Barn [kW]
- Well [kW]
- Living room [kW]

2.1.3 Weather Data

- temperature: actual temperature recorded
- apparent temperature: ”feels like” temperature
- humidity: humidity level of the atmosphere
- pressure: atmospheric pressure
- dewPoint: temperature at which air must be cooled to become saturated with water vapor
- windSpeed: recorded wind speed
- windBearing: recorded direction of wind
- visibility: visibility of the time of day
- cloudCover: state of the sky that is covered by clouds
- precipIntensity: Intensity of precipitation
- preciProbability: Probability of precipitation

2.1.4 Time/Description

- Recorded timestamp (in Unix format).
- icon: Weather conditions in text format
- summary: Another description of weather conditions in text format

2.1.5 Exploratory Data Analysis

When conducting the EDA, we found general trends that were reinforced by common trends found in other houses. For instance, we found that actually the home office contained the

highest energy consumption. This could be explained bny the fact that people will spend extended amounts of time in their home office, leading to a higher expenditure of energy in these rooms.

In terms of the appliances, the microwave and dishwasher had similar monthly energy consumption patterns throughout the year. Interestingly enough, the refrigerator showed a more seasonal trend, with higher energy consumption in the warmer months of the year.

This was in tune with the overall weather trends that we saw in relation to energy consumption. While there was barely any corelation between weather and energy consumption, we saw a positive corelation between energy and temperature. Indeed, the hotter months led to more energy usage, most likely explained by the air conditioning and the refrigerator and other appliances that have to work harder in the warmer months.

2.2 LLM/RAG

2.2.1 Text to SQL

Since the initial approach of fine-tuning an LLM on tabular data by turning each row into a sentence and feeding it to a model was inefficient, coupled with the inaccurate results from basic testing, it made more sense to implement a Text to SQL workflow.

We used LangChain to create a SQL agent, which takes as parameters a large language model and a SQLDatabase, which combine together to make a SQLDatabaseToolkit. The large language model we used is from OpenAI and the SQLDatabase is a PostgreSQL instance. From there, the SQL agent takes as input a text string, converts it into a SQL query, and runs the execution to retrieve the data and complete the calculations.

2.3 Guardrails

2.3.1 Input Moderation

We used NEMO guardrails to ensure that our chat bot interactions remain safe and relevant. We customized the guardrail with a configuration and prompt file. In summary, this input moderation guardrail protected us from:

- Off topic prompts that had no relevance to energy usage and smart homes
- Harmful prompt injections as well as malicious code
- Sensitive data being inputted
- Explicit or garbled language

2.3.2 SQL or Not?

Since our workflow contains two main ”parts” in the form of a Text2SQL workflow and a general LLM workflow; we decided to implement a guardrail that determines whether or

not the question asked by the user can be answered by performing SQL on the data or not. (more information on that later)

2.3.3 RAG Evaluation

(more information on that later)

2.3.4 Output Moderation

(more information on that later)

2.4 LangGraph's Persistence Layer

(more information on that later)

2.5 Product Workflow Diagram

(More info on that later)

3 Results

Below we test the efficiency of our chat bot with some general questions, as well as test each component of the guardrail. We will collect and compile the results by splitting it up into the following sections:

- general chat bot functionality (maybe split into sql and non-sql prompts)
- input moderation
- sql vs non sql guardrail
- rag evaluation guardrail
- output moderation guardrail

3.1 Observations

(more on that later)

3.2 Prompt-to-Output (end-to-end flow) Examples

(more on that later)

4 Discussion

Below we briefly outline some shortcomings (will expand upon later)

- training prebuilt models on our data (very low accuracy)
- using text2SQL for all tasks (some tasks don't work and will cause the model to geek out)
- False denial of service in terms of guardrails

4.1 Fine-Tuning

In our original plan, we explored fine-tuning as a means to specialize the LLM model for our smart home chatbot. The initial strategy was to fine-tune the model (we chose Google-flan-t5 as test model) so that it could effectively "memorize" the smart home dataset, thereby eliminating the need for users to re-enter data for each query. However, we encountered critical challenges that ultimately shifted our approach. After dedicating 9 hours to training, the model only achieved an accuracy of around 40%, underscoring a significant limitation in applying traditional LLM fine-tuning techniques to structured, numerical (tabular) data. This shortfall can be largely attributed to the inherent design of LLMs, which are optimized for processing natural language rather than numerical or tabular information. Moreover, the extensive size of our dataset introduced prohibitive costs in terms of both time and computational resources. These factors collectively led us to pivot toward a more stable solution: Text2SQL pipeline.

4.2 SQL-Suitability of Prompts

(more on that later)

4.3 Guardrails Effectiveness

(more on that later)

4.4 Component Integration

(more on that later)

5 Conclusion

6 Appendix

Digital Bouncers aims to enhance the safety and reliability of AI-driven smart home systems by implementing an LLM-based assistant with robust AI risk mitigation. As smart homes become more integrated with AI, ensuring secure and contextually appropriate interactions is crucial. The proposed solution is a chatbot application that assists users in understanding and managing their smart home energy consumption while enforcing guardrails at input, intermediary, and output levels. We hope to demonstrate the effectiveness of both the chat bot itself, as well as the effectiveness of the guardrails built upon the chat bot.

7 Contributions

- Eric Chen:
  - Performed EDA on energy consumption of appliances
  - Attempted to fine tune pre-trained LLM on the dataset for a more specialized chat bot
  - Tokenized numerical datasets
  - Implemented a working Text2SQL workflow using Lang Chain agents
  - Wrote README file
  - Created first version on prototype of streamlit app
  - looking into making a text2SQL workflow guardrail
- Phu Dang:
  - Performed general EDA on the dataset
  - Implemented a working persistence layer within the LLM models
  - Implemented a Postgres database to store and access conversation checkpoints
  - Created a Postgres instance to store smart home data for Text2SQL workflow
  - Combined Persistence layer with an input NeMo guardrail
  - Completed a working prototype of a streamlit app
  - Created a working docker image
  - Looking into RAG evaluation guardrail
- Daniel Li:
  - Performed EDA on energy consumption in relation to weather patterns
  - Attempted to implement and fine tune LLMs through hugging face APIs
  - Tested different in-prompt data provision methods with different parameters
  - Completed and configured the input moderation guardrail
  - Compiled and Wrote the report
  - Looking into Output Moderation Guardrail and Final Website
- Bofu Zou:
  - Performed EDA on energy consumption based on rooms
  - Contributed to majority of the fine tuning work for LLMs for our dataset

- Tokenized the numerical dataset
- Implemented working text2SQL workflow via Lang Chain agents
- Looking into making a text2SQL workflow guardrail

7.1 Inline Citation Examples

Citation in text (no parentheses): use `\cite{citekey}`. For example, Breiman (2001), Devlin et al. (2019).

Citation in parentheses: use `\citep{citekey}`. For example: (Vaswani et al. 2017), (Karras, Laine and Aila 2019).

To edit the contents of the "References" section, edit `reference.bib`. Many conference websites format citations in BibTeX that you can copy into `reference.bib` directly; you can also search for the paper on Google Scholar, click "Cite", and then click "BibTeX" (here's an example).

References

Breiman, Leo. 2001. "Statistical Modeling: The Two Cultures." *Statistical Science* 16(3): 199–215. [Link]

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics. [Link]

Karras, Tero, Samuli Laine, and Timo Aila. 2019. "A Style-Based Generator Architecture for Generative Adversarial Networks." In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [Link]

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." In *Advances in Neural Information Processing Systems*. [Link]

Appendices

A.1 Training Details

A.2 Additional Figures

A.3 Additional Tables