ChatGPT based ChatBot Application

1st Astha Thapa
School of Computing
Grand Valley State University
Allendale, MI 49401, USA
thapaast@mail.gvsu.edu

2nd Dr. Rajvardhan Patil
School of Computing
Grand Valley State University
Allendale, MI 49401, USA
patilr@gvsu.edu

***Abstract*—Since the invention of ChatGPT, there have been several applications demonstrating how ChatGPT can be used to solve real world problems. This paper focuses on implementing one such application, where we design ChatBot using ChatGPT API. To carry out the experiments, we choose dataset from a specific domain, and store the data in graph oriented database. Sample training dataset consisting of English and its equivalent Cypher queries were designed for prompt tuning ChatGPT on the local Knowledge-graph. This training dataset helps ChatGPT understand the mapping between the English and their equivalent Cypher queries. Once the mapping is learnt, it becomes easier for ChatGPT to encode or represent a real time (previously unseen) English query fed by the user into its equivalent Cypher query format. In this paper, we demonstrate our results on sample Rope-related dataset scrapped from internet, and conclude with limitations and future work.**

***Index Terms*—Chatbot, Chat Assistant, ChatGPT API, Knowledge Graph, Cypher Query Language, Graph Database**

I. Introduction

In recent years, there has been growth in the demand for Large Language Models (LLMs) due to their ability to understand, summarize, and generate data similar to a human being. This transformative impact of LLMs across multiple domains has allowed LLMs to be in a position of one of the most powerful and cutting-edge tools for automation. As more and more businesses and industries are recognizing the potential of LLMs, there has been a rising trend of integrating them into day-to-day applications. In layman's terms, Large Language Models (LLMs) are specialized type of artificial intelligent models that are designed to have the ability to understand human language and recreate a human-like sense of the language in a remarkably natural way. The emergence of LLMs such as ChatGPT has allowed many people to know about its impressive capabilities, significant power, and impact. As discussed in Related Work (Section V), LLMs can be used across various application domains to carry out Natural Language Generation (NLG) and/or Natural Language Understanding (NLU) tasks. We explore the NLG ability of the model especially related to generation of Cypher queries, where the idea is to make the Chat assistant respond back to User queries with results that are domain sensitive/specific. The reasons for choosing LLMs to perform the mapping between English and Cypher queries were threefold:

1) Language Translation: The ability to accurately translate languages along with the ability to be contextually aware makes LLMs one of the best tools for Language Translation applications.
2) Code Generation: LLMs can be used in code generation as they assist developers in writing complex and efficient programs. One good example is the Github co-pilot which acts as a code-writing assistant.
3) Chatbots or chat assistance: One major application of LLMs is to enhance the user experience by allowing users to interact with the system via chatbot.

While LLMs are great tools, designing them from scratch incur humongous amount of time, energy, and effort. We therefore opted for accessing LLM using API, and then prompt tuning them using specific dataset. We wanted to select dataset that was very unique and not explored before, and had less chance of been used during the pretraining of LLMs. After extensive search, we decided to chose 'Ropes' related dataset. To store this domain specific knowledge of 'Ropes' we used Graph Database (Neo4j), and constructed sample English questions with their Cypher query equivalent.
The outline of this paper is as follows. In section II, we describe the reasoning for using Cypher query and Graph database for this work. The metadata of knowledge graph along with sample Cypher queries are presented in Section III. In Section IV, experiments and results of the proposed prototype are highlighted followed by related work in Section V. Section VI elucidates the possible future work, with Conclusion in Section VII.

II. Databases and Query Languages

To store the domain specific data, instead of using structured Realtional Database (RDB), we opted for graph database (Neo4j), since the query language of Graph database (CypherQL) seems to have more proximity to English query language as compared to RDB's Structured Query Language (SQL). Additionally, unlike the tables in RDB that are connected with primary-foreign key, in Graph Database, labels can be used to distinguish different types of relationships. These labels also help in easy formulation of queries while retrieving the data. Also, unlike SQL queries which can become complex at times especially when more tables are involved, Cypher queries are easier to expand and less complicated even when there are more nodes to be traversed in the query. Compared to the computationally expensive joins from RDB's Structured Query Language

SoutheastCon 2024 | 979-8-3503-1710-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/SOUTHEASTCON52093.2024.10500264

(SQL), traversing and fetching the desired 'rope' data from graph database was relatively faster.
Furthermore, when it comes to query structure, cypher queries are much more easier to construct as they are more relatable and closer in proximity to English queries in terms of syntax and structuring, allowing users to query graph data in flexible manner. For example, let's assume we have the following nodes in Neo4j database:
$(Sam) - [: NEIGHBOR\_OF] -> (Bob)$
$(Sam) - [: NEIGHBOR\_OF] -> (Jim)$
$(Jim) - [: NEIGHBOR\_OF] -> (Mike)$
Now if we have to find out all the neighbors of 'Sam', we formulate the following query:
$MATCH (Sam) - [: NEIGHBOR\_OF] -> (neighbor)$
$RETURN neighbor.name;$

To extend the example further, if we had to find 'Who are Sam's neighbors that have a dog?', the query can easily be expanded to:
$MATCH (Sam) - [: NEIGHBOR_OF] -> (neighbor) - [: HAS] -> (dog : Dog)$
$RETURN neighbor.name;$

Such ease of conversion between English and their equivalent Cypher queries along with better data representation, leads to better accuracy and performance gain. Overall, when it comes to representing real world entities and connection between them, graph database and its query language are comparatively better choice, as they are much more flexible and expressive in establishing network connections as well as capturing their relationships using labels.

III. Methodology

A. Knowledge Graph Metadata

ChatGPT is a great tool for getting information while maintaining a sense of conversation which makes it easier for humans to interact with. However, one downside is that it can "hallucinates" - meaning that sometimes it may provide with the wrong information. In the client-to-business environment, even a tiny bit of wrong information can have catastrophic results, such as losing some porition of customer base. This is why one of the major focuses of this project was to keep the GPT inbound within the scope of the business domain by using the right context. To make this possible, knowledge graph that represents the business domain knowledge was constructed using Neo4j. Neo4j is an open-source Graph Database that allows the data points to be represented as nodes, where they can be connected using labels if required. For example, to represent availability of Rope in certain specification, we can use 'Details' node, and connect it with 'Rope' node via 'Available_IN' relationship label shown below in Figure 1.

The reason for using Neo4j was because of its flexibility in the representation of the data model in terms of nodes and relationships. Once the ChatGPT model is aware of the nodes and their relationships, it becomes easier for it to convert English queries into their respective Cypher queries.

Fig. 1. Node Relationship

As highlighted in Figures 2 and 3, the 'Rope' node contains information such as the 'name, color, description, highlights, material, title etc.' of the rope, and the 'Details' node contains information about the attributes of the rope such as 'length, size, diameter, strength etc.'.

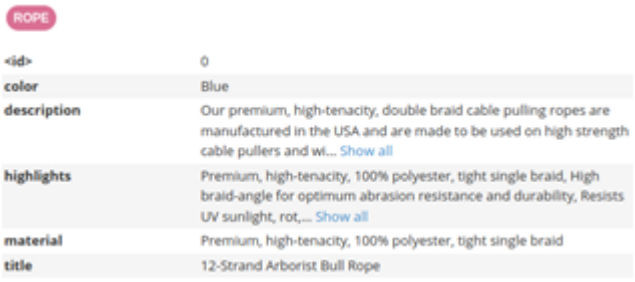

Fig. 2. Rope Node

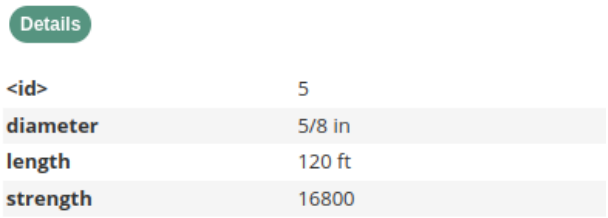

Fig. 3. Details Node

The reason for creating Nodes using such metadata or schemas was because that way the Rope Node is unique to a single rope while the information of the Detail Node can be shared by multiple Rope nodes. So, the main idea was to create nodes that would establish a sensible relationship aligning with the user's query about the business domain. For this project, we collected data on six different types of rope. Tables I and II give the details of rope dataset metadata and relationship information. In total there were a total of 296 Nodes. The following table showcases the relationship between each Nodes.

B. Cypher Queries

After creating the knowledge-graph database, the next step was to create sample English and their equivalent cypher queries for prompt tuning purpose. Prompt tuning is much faster and efficient as compared to fine tuning approach, since

TABLE I
ROPE DATASET METADATA

Node Name	#Nodes	Properties
ROPE	6	Color, Description, Highlights, Material, Title, URL
Details	137	Diameter, Length, and Strength
Reviews	16	Price of Quantity 1, Quantity 2, Quantity 3 and Quantity 4
Price	137	Comment, Stars, and username

TABLE II
RELATIONSHIP INFORMATION

Node	Relationship Type	Node
Rope	AVAILABLE_IN	Details
Details	Cost_Details	Price
Rope	ratings	Review

the model demands significantly much lesser data during prompt tuning. Using the sample mapping of queries, during prompt-tuning, ChatGPT understands the context and domain knowledge, which enables it to map an unseen English query to its equivalent Cypher query.

Cypher Query is a graph query language that allows to retrieve data from the Neo4j Graph. Following is a basic Structure of a cypher query.

$MATCH(: Entityentity_attributescriteria) - [: Relationship]->(AnotherEntity)$
$RETURNResults$

For example, to retrieve a Person named "John" who reads "fiction books", the following would be the cypher query.

$MATCH(person : Personname : "John") - [: READS]->(book : Bookgenre : "fiction")$
$RETURNperson, book$

We constructed, the mapping between English query and their equivalent Cypher query for almost 70 queries. These queries were then fed as input to the ChatGPT through the prompt for tuning purpose. Below are some examples of such mapping-pairs.

1) What is the material used for 12-Strand Arborist Bull Rope ?
   MATCH (rope:ROPE) WHERE rope.title CONTAINS "12-Strand Arborist Bull" RETURN rope.material
2) Can you tell me about Wonder Rope – Arborist Bull Rope?
   MATCH (rope:ROPE) WHERE rope.title CONTAINS "Wonder Rope – Arborist Bull" RETURN rope.description
3) What are available colors for Anchor Lines ?
   MATCH (rope:ROPE) WHERE rope.title CONTAINS "Anchor Lines" RETURN rope.color
4) Can you get me the description of the rope with the title "Double Braid Mooring Pendant"?
   MATCH (rope:ROPE) WHERE rope.title CONTAINS "Double Braid Mooring Pendant" RETURN '{key:'+rope.title+', value:[' + rope.description+ ']}'
5) Find the user who left a review for the rope with a diameter of 3/8 in and a length of 100 ft
   MATCH (review:Reviews)-[:rating]->(rope:ROPE) MATCH (rope)-[:AVAILABLE_IN]->(details:Details) WHERE details.diameter CONTAINS "3/4" AND details.length CONTAINS "100 ft" RETURN COLLECT(review.user) AS users
6) Retrieve the materials and highlights for the rope with the highest rating.
   MATCH (rope:ROPE)<-[:rating]-(review:Reviews) WITH rope, review ORDER BY review.stars DESC LIMIT 1 RETURN rope.material, rope.highlights
7) Find the ropes that have reviews with comments containing the word "durable".
   MATCH (rope:ROPE)<-[:rating]-(review:Reviews) WHERE review.comment CONTAINS "durable" RETURN rope.title, review.comment
8) What are the diameters of the ropes available in a length of 100 ft?
   MATCH (rope:ROPE)-[:AVAILABLE_IN]->(details:Details {length: "100 ft"}) RETURN COLLECT(DISTINCT details.diameter) AS availableDiameter

The following is a code snippet for calling the GPT API. Here, query is the part where we send in the sample dataset for prompt-tuning purpose.

$constresponse = awaitopenai.createCompletion(\{$
$model : "text - davinci - 003",$
$prompt : query,$
$temperature : 0,$
$max_tokens : 150,$
$top_p : 1.0,$
$frequency_penalty : 0.0,$
$presence_penalty : 0.0,$
$stop : ["\#", ";"],$
$\});$

While implementing this approach, there was a cost associated to use the ChatGPT API. There is a character limit to how much data can be send through the prompt, and each such API request incurs cost, which is based on the number of words that are being sent (along with the response's word count).

IV. SYSTEM OVERVIEW

In contrast to the proposed system leveraging OpenAI's API, traditional chatbots that do not integrate or leverage the AI technology have significant limitations. These types of chatbots rely on rule-based approaches and lack a sense of dynamism in the conversation. These in return impact the user experience as the user experience tends to feel rigid and scripted. Additionally, users are often constrained by predefined and limited interaction options and are often

left with restricted user experience. The absence of an AI integration into a chatbot hampers the ability of the system to deliver engaging and personalized experiences. Whereas, a chatbot leveraging OpenAI's API significantly reduces the time and resources required for setup and implementation, and allows user-interactive chatbots to be developed swiftly across various domains.

This section delves into the architecture of the proposed AI-integrated system. The system architecture components of the proposed approach are as follows: Figure 4 showcases the workflow involved in executing the English query by our proposed prototype.

1) Graph (Database) Component: The graph database is used for storing the information of the system that the user wishes to know. It is similar in concept to the one that is used in most commercial business environments. This component of the system gives the system the ability to remember the information and retrieve it when necessary. The proposed system stores information related to "Rope" such as types of ropes, attributes of each rope type, and their related prices. This could be potentially replaced by any other type of database such as a relational database, graph databases possess many flexible advantages over a traditional relational database.
2) GPT (LLM) Component: The other component of the system is GPT, a LLM component. As mentioned several times in this paper, GPTs are great tools that can understand Human language to a great extent. The proposed system has accessed the GPT from an API. Unlike its infamous usage of giving results from its knowledge base, the proposed system uses the "understanding ability" of GPT allowing the dynamic creation of executable cypher queries based on the user query which is in English.
3) Application Component: The application component of the system is the one that integrates all of the components. It mainly contains the UI part, where the user interacts, and the backend part which is responsible for sending in a prompt to the GPT takes that prompt and executes it in the database for retrieving the information.

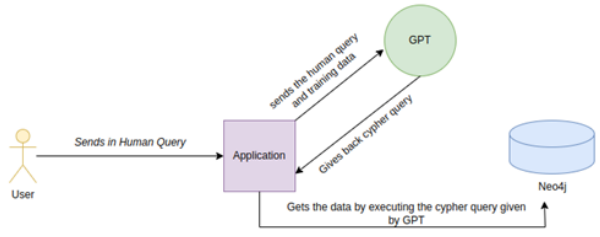

Fig. 4. Workflow

For most LLMs, one would require training and pre-training the model to produce this kind of result with such high accuracy. However, GPT allows the usage of its API in such a way that a small amount of text sent as a Training dataset results in better accuracy of the results. The proposed system has created a sample user query and its equivalent cypher queries that will be sent in as a training dataset. However, unlike the traditional way of pre-training the LLM, for GPT one must always send the training dataset with each prompt it sends to the GPT model. The limitations of such an approach will be discussed in detail in the Limitation Section of this paper.

V. Experiments and Results

As shown in Figure 5, we used a simple UI in React that corresponds to a chatbot. The user will send in their query (a normal sentence in English) through the application's chatbot UI. The application then sends the human query along with prompt data to the ChatGPT. ChatGPT returns back a cypher query equivalent to the user's English query. The application then sends the query to Neo4j database, which executes the query and returns the result back to the user.

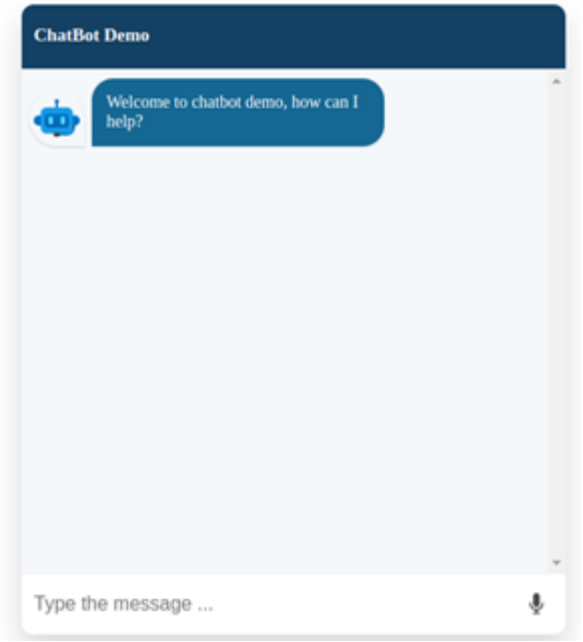

Fig. 5. Chatbot UI

When user enters an English query, for instance "What kind of ropes do you have or sale?", the system receives the following translated Cypher query from the ChatGPT API.
$\backslash MATCH(rope : ROPE)$
$RETURN\{ropeTitle : rope.title, description : rope.description\}ASresult$
This query is then executed in the local neo4j server. As shown in Figure 6, the response from neo4j is then forwarded to the user interface as the query result to their original English query.

A. Advantages

The major advantages of our proposed approach are:

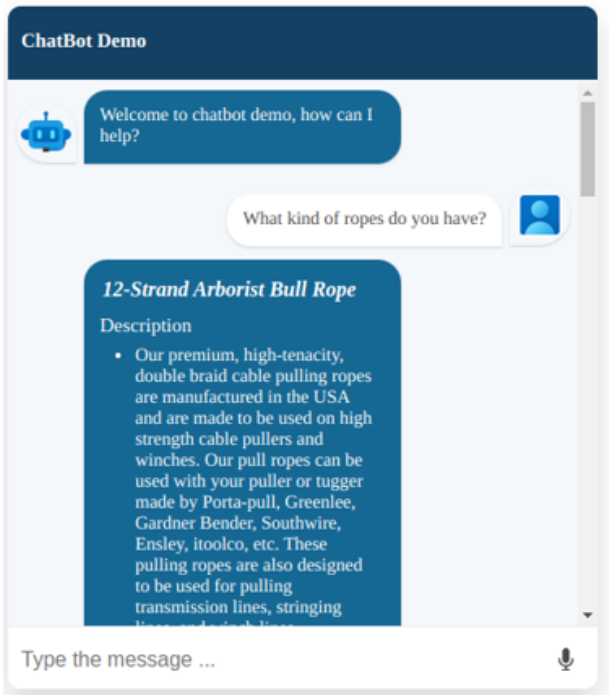

Fig. 6. Chatbot response

1) English or Natural query language can be used to interact with the system.
2) Eliminates the need to learn a query language to interact with the database.
3) No need to collect huge dataset, as sample dataset is sufficient for prompt tuning the model via API.

B. Limitations

While this approach offers several advantages, the following limitations must be acknowledged to get a comprehensive understanding of the implementation. We need to recognise and address the inherent limitations associated with this approach to fully grasp both the capabilities and constraints of the system. Some of the limitations this sections explored in this section are as follows:

1) Inability to handle incorrect queries: One major area for improvement of this approach arises when the users ask incorrect queries. While the system is created with the vision of adapting to the dynamic questions of the users and fetching appropriate information based on that, it would need help to answer queries that are outside of the knowledge domain. OpenAI's GPT model is powerful enough to provide an answer -based on its knowledge base, but it will fail to construct a proper cipher query expected by the code, leading to errors.
2) Need to Send Training Dataset with Every Prompt: One major limitation that leads to subsequent limitations in this approach is the need to send the training dataset with every prompt. While this acts as a quick way to train the GPT to provide us with the output as per our need, it also means a limitation on further fine-tuning the responses and an increase in cost.
3) Associated Cost: Open AI is a wonderful tool that has allowed a chain of technological advancement in the field of software engineering but it comes with an associated cost. The cost is directly related to the number of tokens sent as prompt and the number of tokens received as output. So one must evaluate the cost-to-benefit ratio while implementing this approach.
4) Limit on number of tokens sent: Along with the associated cost, the other limitation is the restriction on the number of tokens sent. This greatly imposes restrictions on the training dataset sent as a prompt. Since you must always send training datasets along the prompt for the GPT to understand the context of what is expected as output, the implementors must be very careful about creating a training data set that would not only capture the essence of the expected output but also not exceed a certain number. This also includes the user query which is not always predictable.
5) Requirement of knowledge graph: To keep everything within our knowledge base, this approach utilizes the implementation of the knowledge graph. However, the complexity of the query linearly increases if the information to be represented is complex, which means the more complex a relationship exists in the knowledge graph, the more complex queries the GPT must be able to produce.

C. Mitigation

While the above-mentioned are mostly related to the functioning of OpenAI's API, a few of the limitations are on the approach itself. The following section discusses the mitigation of these limitations.

1) Handling incorrect queries Since the UI space is a chatbot, the user can structure the question however they want. This leads to several errors. One way to mitigate this situation is to use an exception handler in the cypher query execution. Any response returned by the GPT that is not a Neo4j cypher query will cause an error while being executed in the database and hence a default response can be sent in as a query response to the user.
2) Set a limitation on the user input The number of tokens being sent as a prompt is known to us, as it is something that has been created on our end. However, the user's input which will be eventually sent in as an input token is open-ended currently in this approach. Setting a limitation on the user input from both the front end and back end will help us determine the maximum number of user input tokens which will help us have a sense of the number of tokens being exchanged and the associated cost with that.

VI. Related Work

In this section, we will talk broadly about works that have utilized a common approach such as utilizing the ChatGPT

API to solve various real world problems. Below, we classify these examples into eleven different categories.

A. Education

[1] is a good example of using the ChatGPT API in the context of a library management system - Aisha. The authors have demonstrated how they have been able to create a custom AI library chatbot using chatgpt API for Zayed University Library. They have used a vector database to store the information and used LangChain to create their context and queries which they can send to Open AI. [4] has been able to demonstrate the concept of utilizing ChatGPT to narrow the domain through EduChat, a chatbot system that answers all questions related to the university. However, they have been able to create a hybrid system where if the queries are within the context of the application, predefined answers are returned else queries are sent to chatGPT API with proper context. [6] shows the usage of ChatGPT to create an educational environment where the custom chatbot plays the role of the teacher. Through interactive educational conversations, the chatbot was able to cover multiple learning topics related to social media literacy.
[15] discusses the use of chatGPT OpenAI through an automation Middleware where the authors of the paper integrated it into Microsoft Teams to act as a tutoring chatbot. The chatbot's functioning could be triggered with the keyword "Kiran". [18] explores the potential of using OpenAI's API in the domain of start-up ideas generated by students. The paper proposes a prototype where students can refine their start-up idea through interactions with a customized chatbot that would provide them with diverse feedback which will allow them to refine their start-up idea iteratively. [21] proposes various kinds of solutions to solve university-student interaction problems where if students were left to interact directly with a free chat, it could even end up in legal issues if answered incorrectly. This paper showcases how a custom chatbot created with university rules and data leads to controllable dialogues.

B. Multimedia

[2] showcases how ChatGPT has been integrated into the field of animation production. With the combination of Unity plug-ins and ChatGPT API, they have been able to come up with an animated environment where the user can interact with a digital human. [3] discusses the usage of ChatGPT in aiding vision-centric tasks such as selecting an object in the image through mouse clicks and/or pointing at them. The authors of the paper have created InternGPT which stands for Interaction Nonverbal Chatbots. An example discussed in the paper is the comparison between only textual commands to the chatbot versus the usage of pointing-language-driven interactive systems and textual commands to remove an object from the image.

C. Legal Field

[5] leverages ChatGPT API in their chatbot to provide reliable support 24/7 by utilizing publicly available legal documents. In order to overcome the token limit sent in the API, they used LlamaIndex through vectorization.

D. Software development

[7] does not focus on creating a custom chatbot through ChatGPT API but it still talks about the usage of a Language model in the field of software development specifically Dev Security Operation. It talks about how they used the GPT API to figure out the flaws in their IaC scripts through the analytical power of chatGPT.

E. Healthcare

[8] presents the utilization of ChatGPT and GPT-4 in the field of data processing and de-identification of medical data. The authors of this paper were able to showcase examples in which they were able to accurately adhere to the HIPAA rules governing medical data. Authors of article [12] investigated ways to improve the performance of a chatbot using ChatGPT. They developed a system called "OntoChatGPT" and demonstrated how it could be used in rehabilitation using the Ukrainian Language. [14] dives into the usage of the ChatGPT model in the field of urology. The authors of this paper used ChatGPT API with a Python script to come up with a system that correctly answers queries related to the urology department in Natural language. [20] exhibits how GPT3's API and OPT350m models can be used to generate medical reports based on the COVID-19 CT scan of patients. The authors of the paper concluded that the report generated had good terminology prediction.

F. Statistics

Authors of [9] demonstrated the creation and usage of a custom chatbot named "ChatSQC" which stands for Statistical Quality Control. They focused on creating a tool that is capable of answering precise SQC-related questionnaires.

G. Science and Engineering

[10] explains the usage of chatGPT API in the field of chemistry where the authors were able to create a custom chatbot called MOF, a chemistry assistant that would answer questions on chemical reactions and synthesis procedures. [14] introduces the integration of chatGPT with Rasa - an open-source framework for developing task-oriented chatbots. They designed Rasa custom actions that would invoke OpenAI's API in order to process complex customization rules and hold a conversation in a smart environment. [16] illustrates the proof of utilizing the ChatGPT API with ROSGPT to create a robotic system that runs on the user command. It elaborates on the transformation of human language to robotic instruction.

H. Game Development

[11] explores the idea of integrating the power of ChatGPT in the field of Game development. A debate game had been created where the users would be able to have debates through the customized chatGPT responses with different levels of difficulty along with an evaluation system.

I. Business

[13] narrates the usage of empowering customer services in the business environment through the integration of chatgpt in a customized chatbot. It dives deeper into the process of creating an expert system powered by OpenAI's API along with various applications. [17] explains the usage of chatGPT API in a JBMS chatbot in order to improve the user experience by focusing on the quality of interactions of the users. The chatbot not only provided a variety of information on the searched articles but also provided users to directly submit an article page.

J. Language and Translation

[19] addresses the usage of chatGPT API in the field of language and translations where the authors of the paper came up with an idea of integrating the capabilities of GPT with voice assistance supporting multilingual functionality.
Our work focuses on using ChatGPT API to design Chatbot that can assist users in retrieving relevant information from a specific knowledge domain graph.

VII. Future Work

We were able to successfully design and implement the Chabot prototype with basic functions. However, we plan to extend this prototype system by incorporating the following features:

1) Utilizing NLP Pipeline for Knowledge Graph Generation: Instead of manually creating the Knowlede-graph, we can leverage Natural Language Processing (NLP) pipelines to automate the generation of this knowledge graph. We plan to explore whether we can efficiently extract and structure data using NLP, allowing for the automatic creation and maintenance of a comprehensive knowledge graph, which could save time and resources.
2) Enhancing Response Quality with NLP: In the current project setup, the application sends the Cypher queries received from ChatGPT to the local Neo4j server, to execute and retrieve the results. The user interface then displays these responses, typically in the form of lists of answers. While this fulfills the initial prototype requirements of providing answers to queries, it falls short of creating a truly conversational experience. To improve this aspect, we plan to implement NLP methods to make responses more human-like and conversational. One approach involves sending the responses back to NLP model and asking for assistance in rephrasing the answers to make them sound more natural and conversational. This could significantly enhance the user experience, making interactions with the system feel less robotic and more human-like, ultimately improving the overall usability and appeal of the application.
3) Designing Generic Template: We plan to extend our work to build a Chatbot template, where users can simply provide database and sample queries as input to the Template. Such template will be generic and flexible to use, and will not be limited to any specific dataset or domain.
4) Incorporating time and accuracy conversation: As a future work, we plan to analyze the accuracy of the conversation along with the time required for the chatbot to create the response. This will help to provide a good robust comparative study supported by quantitative data.
5) Incorporating user feedback: As the proposed system targets a real-world audience, the next logical step that can be added is incorporating the user into the equation as a future endeavour. This will help us understand what areas need analyzing and improvement, particularly in terms of dynamic interactions

VIII. Conclusion

This paper demonstrated how ChatGPT can be used in developing Chatbots that assist users in finding the desired information. Our approach involved scrapping data from internet and storing them in graph oriented database. Later, sample Cypher-queries along with their English counterparts were passed through API to ChatGPT. This sample training dataset helped ChatGPT understand and correlate the mapping between English and Cypher queries for the given dataset. In addition to the training dataset, the user query was also injected as part of the context for which ChatGPT returned its equivalent Cypher query. This Cypher query was then executed on the knowledge Graph to generate the output and display it to the end user.

References

[1] Lappalainen, Y., & Narayanan, N. (2023). Aisha: A Custom AI Library Chatbot Using the ChatGPT API. Journal of Web Librarianship, 1-22.
[2] Lan, C., Wang, Y., Wang, C., Song, S., & Gong, Z. (2023). Application of ChatGPT-Based Digital Human in Animation Creation. Future Internet, 15(9), 300.
[3] Liu, Z., He, Y., Wang, W., Wang, W., Wang, Y., Chen, S., ... & Qiao, Y. (2023). Interngpt: Solving vision-centric tasks by interacting with chatgpt beyond language. arXiv preprint arXiv:2305.05662.
[4] Dinh, H., & Tran, T. K. (2023). EduChat: An AI-Based Chatbot for University-Related Information Using a Hybrid Approach. Applied Sciences, 13(22), 12446.
[5] Qasem, R., Tantour, B., & Maree, M. (2023). Towards the Exploitation of LLM-based Chatbot for Providing Legal Support to Palestinian Cooperatives. arXiv preprint arXiv:2306.05827.
[6] Koyuturk, C., Yavari, M., Theophilou, E., Bursic, S., Donabauer, G., Telari, A., ... & Ognibene, D. (2023). Developing Effective Educational Chatbots with ChatGPT prompts: Insights from Preliminary Tests in a Case Study on Social Media Literacy. arXiv preprint arXiv:2306.10645.
[7] Petrović, N. (2023). ChatGPT-Based Design-Time DevSecOps. Preprint.
[8] Liu, Z., Yu, X., Zhang, L., Wu, Z., Cao, C., Dai, H., ... & Li, X. (2023). Deid-gpt: Zero-shot medical text de-identification by gpt-4. arXiv preprint arXiv:2303.11032.
[9] Megahed, F. M., Chen, Y. J., Zwetsloot, I., Knoth, S., Montgomery, D. C., & Jones-Farmer, L. A. (2023). AI and the Future of Work in Statistical Quality Control: Insights from a First Attempt to Augmenting ChatGPT with an SQC Knowledge Base (ChatSQC). arXiv preprint arXiv:2308.13550.
[10] Zheng, Z., Zhang, O., Borgs, C., Chayes, J. T., & Yaghi, O. M. (2023). ChatGPT Chemistry Assistant for Text Mining and Prediction of MOF Synthesis. arXiv preprint arXiv:2306.11296.
[11] Lee, E. Y., il, N. G. D., An, G. H., Lee, S., & Lim, K. (2023, August). ChatGPT-Based Debate Game Application Utilizing Prompt Engineering. In Proceedings of the 2023 International Conference on Research in Adaptive and Convergent Systems (pp. 1-6).

[12] Palagin, O., Kaverinskiy, V., Litvin, A., & Malakhov, K. (2023). OntoChatGPT Information System: Ontology-Driven Structured Prompts for ChatGPT Meta-Learning. arXiv preprint arXiv:2307.05082.
[13] Manolitsis, I., Feretzakis, G., Tzelves, L., Kalles, D., Katsimperis, S., Angelopoulos, P., ... & Varkarakis, I. (2023). Training ChatGPT models in assisting urologists in daily practice. Stud Health Technol Inform, 305, 576-579.
[14] Gallo, S., Malizia, A., & Paternò, F. Towards a Chatbot for Creating Trigger-Action Rules based on ChatGPT and Rasa.
[15] Rajala, J., Hukkanen, J., Hartikainen, M., & Niemelä, P. (2023, October). Call me Kiran ChatGPT as a Tutoring Chatbot in a Computer Science Course. In Proceedings of the 26th International Academic Mindtrek Conference (pp. 83-94).
[16] Koubaa, A. (2023). ROSGPT: Next-Generation Human-Robot Interaction with ChatGPT and ROS.
[17] Santoso, G., Setiawan, J., & Sulaiman, A. (2023). Development of OpenAI API Based Chatbot to Improve User Interaction on the JBMS Website. G-Tech: Jurnal Teknologi Terapan, 7(4), 1606-1615.
[18] Ilagan, J. B. R., & Ilagan, J. R. (2023). A prototype of a chatbot for evaluating and refining student startup ideas using a large language model.
[19] Ayyash, A. A., & Bharathi, P. D. (2023). Artificial Intelligence based Assistant with Specialization in a Given Language. International Journal of Research in Engineering, Science and Management, 6(5), 68-71.
[20] Mehboob, F., Malik, K. M., Saudagar, A. K. J., Rauf, A., Jiang, R., Khan, M. B., & AlTameem, A. (2023). Medical Report Generation and Chatbot for COVID_19 Diagnosis Using Open-AI.
[21] Bieletzke, S. (2023). AI-CHATBOT-INTEGRATION IN CAMPUS-MANAGEMENT-SYSTEMS. In EDULEARN23 Proceedings (pp. 3574-3583). IATED.