Case Studies on LLM Centric and Services Oriented Data Analytics Agent Development

Hong Qing Yu*
School of Computing
University of Derby
Derby, Derbyshire, United Kingdom
h.yu@derby.ac.uk

Jack Sutton
School of Computing
University of Derby
Derby, Derbyshire, United Kingdom
J.sutton@derby.ac.uk

Sam O'Neill
School of Computing
University of Derby
Derby, Derbyshire, United Kingdom
s.oneill@derby.ac.uk

Stephan Reiff-Marganiec
School of Computing
University of Derby
Derby, Derbyshire, United Kingdom
s.reiff-marganiec@derby.ac.uk

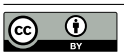
Abstract

This paper presents a novel service orchestration framework for a chatbot application focused on data analytics questions. The framework integrates Large Language Models (LLMs) with service-oriented computing to transform data analytics into a dynamic, conversational experience. The approach leverages advancements in LLM technology to enable real-time, automated data insights via chatbot interfaces, making complex data analytics accessible across various industries. In addition, the data will be processed and analysis at edge-machine rather than post all the data directly to the LLMs on the cloud. Therefore, the Central to the framework is the local Micro Analytics Service (MAS) and a dynamic service-data coordination framework, which together facilitate the decoupling of data from business logic, allowing for intuitive engagement with analytics processes. Through two case studies, retail data analysis and regional healthcare planning, the ability of the framework to provide actionable insights through natural language prompts is demonstrated, showcasing its potential to significantly reduce barriers to sophisticated data analytics. The evaluation reveals strong performance in data connection and code generation, with identified areas for improvement in visualizations and handling complex data scenarios.

Keywords

LLM-driven service orchestration, Dynamic data analytics services, Services Computing

ACM Reference Format:
Hong Qing Yu, Jack Sutton, Sam O'Neill, and Stephan Reiff-Marganiec. 2024. Case Studies on LLM Centric and Services Oriented Data Analytics Agent Development. In *2024 13th International Conference on Software and Information Engineering (ICSIE) (ICSIE 2024), December 02–04, 2024, Derby, United Kingdom.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3708635.3708655



ICSIE 2024, Derby, United Kingdom

ACM ISBN 979-8-4007-1776-5/24/12
https://doi.org/10.1145/3708635.3708655

1 Introduction

Dynamic data analytics, complemented by chatbot conversations, is not only crucial but also increasingly demanded by many businesses in different industry sectors [1]. It becomes much more feasible due to the rapid advancements in Large Language Model (LLM) technology. This approach is essential in services computing and business analytics, particularly given the complexity and rapid evolution of data. Dynamic analytics facilitates real-time analysis and interpretation of continuously updating data, a necessity in today's fast-paced business world where the immediacy and relevance of data are paramount. Traditional static analysis methods are inadequate for such constantly changing datasets, highlighting the importance of dynamic approaches for informed decision-making in areas like market trends, customer feedback, planning and operational optimization.

In addition, the synergy of dynamic data analytics and chatbot interfaces marks a significant advancement in business analytics. It addresses the challenges of managing and interpreting rapidly evolving data and makes these insights accessible and actionable through intuitive, conversational interactions. This innovative combination promises to revolutionize how businesses engage with their data, making complex data analysis more approachable and relevant across various business sectors.

In the realm of services computing for business analytics, the challenge extends beyond the mere integration of services. It involves the handling of diverse data types, spanning various repositories and formats, each representing unique dimensions of business intelligence. This complexity is not just about volume but also about the variety and velocity of data, encompassing structured databases, unstructured text, real-time streams, and more. These disparate data forms, housed in different repositories, offer multifaceted views of business operations and decision-making interactions. Efficient navigation and integration of these diverse data dimensions is crucial to generate actionable insights and driving informed decision-making [2, 3]. Traditionally, the challenge of managing or coordinating diverse business services working dynamically in the data has been approached through semantic web service technologies, such as OWL-S (Web Ontology Language for Services) [4] and RESTful service description ontologies [5].

These technologies were designed to provide a standardized way to describe the functionalities, data requirements, and other characteristics of web services. OWL-S, for instance, enabled the detailed annotation of web services, facilitating the automation of service discovery, invocation, and composition. Similarly, RESTful service description ontologies were employed to describe RESTful APIs, making them more discoverable and understandable. These semantic technologies aimed to create a more interoperable and efficient ecosystem for web services, essential for complex data analytics tasks. However, there are three main issues:

- The adoption of semantic approaches or registration frameworks incurs substantial engineering costs, necessitating specialized programmers to develop and manage semantic data and descriptions, such as with OWL-S systems. It also requires the use of unique query languages like SPARQL for effective searching and selection, alongside intricate requirements and specifications schema, and comprehensive workflow management to accomplish complex tasks. This challenge parallels the industry's transition from Semantic Web to Knowledge Graph technologies, underscoring a preference for simpler solutions over more complex ones [9]. This insight was reinforced through our experience in a project focused on developing an automatic semantic machine learning microservice [10].
- Services designed within these frameworks are typically constrained to perform specific tasks, tethered to a unique data source identified during the design phase. For instance, a service engineered to deliver weather forecasts based on a postcode must be explicitly linked to a designated data source from its inception. Integrating the service's data and business logic becomes a necessity. Altering the data source for the service requires either a complete rebuild of the service to accommodate the new dataset or the introduction of an additional data adaptation service. This adjustment must be integrated beneath the operational service within a static composition workflow, which complicates the flexibility and scalability of the service.
- There remains a significant gap between the service layer and the user interaction layer in traditional web service architectures. These architectures typically focus on data transformation for the interaction layer or backend task completion. This structural separation poses challenges to the seamless implementation of chatbot-style interactive applications, which require more integrated and user-centric approaches to service design and delivery.

The advent of sophisticated AI-driven methodologies, including Large Language Models (LLMs), heralds a paradigm shift toward more dynamic and context-sensitive solutions. These advanced systems are designed to seamlessly integrate and scrutinize data from various sources, effectively surpassing the constraints associated with traditional semantic web approaches.

In this paper, we unveil three key contributions that stem from our collaborative research with industry partners:

- The introduction of a novel service concept, termed Micro Analytics Service (MAS), which is an algorithm-based service. Unlike traditional services, MAS operates independently of data sources, placing a greater emphasis on business logic and the data analytics process itself.
- The development of an LLM-centric dynamic service and data coordination framework, specifically engineered to facilitate chatbot-enabled data analytics applications. This framework represents a significant leap forward in harnessing the power of conversational interfaces for complex data analysis tasks.
- A strategic approach to data processing and analysis that prioritizes edge or local environments over direct cloud-based processing with LLMs. This method stands in contrast to existing frameworks, such as PandasAI, by offering a more localized and potentially more secure way to handle data [11].

Towards the conclusion of our discourse, we will showcase two case studies alongside evaluation benchmarks that highlight the applicability and effectiveness of our contributions within the contexts of retailer data analysis and regional healthcare planning scenarios. These practical applications not only illustrate the potential of our proposed solutions but also their versatility and impact across different domains.

2 Related Work

Service composition in the ChatGPT era [12] discusses the potential of Large Language Models (LLMs) like ChatGPT to automate service composition in Service-Oriented Computing. It highlights the capabilities of ChatGPT in service discovery, composition, and interface interpretation, presenting a significant shift towards automation in service-oriented systems with minimal supervision. The article identifies challenges such as prompt engineering, composition verification, and execution monitoring as future research directions. It suggests that LLMs can elevate automated service composition to new levels of autonomy, but emphasizes the need for further research and development to address existing limitations and ensure reliability.

JarviX: A LLM No-code Platform for Tabular Data Analysis and Optimization introduces JarviX, a platform employing LLMs for seamless tabular data analysis and optimization, emphasizing trivial yet crucial services like data cleaning and ETL operations, which are standardized for any tabular dataset [13]. It highlights the platform's capabilities in automated, high-precision analysis and integrates AutoML for predictive modeling, forming an automated optimization cycle. The paper also discusses improvements in LLM personalisation, data type support, and user interface improvements for a more intuitive experience. Provides a comprehensive examination of the role of smaller services (similar to our proposed MAS) in data preprocessing, making advanced data analysis techniques more accessible to users without deep technical expertise.

DataFrame QA: A universal LLM framework for answering data frames without data exposure offers a novel framework for interacting with data frames using natural language queries while ensuring that sensitive data are not exposed [14]. The article highlights the methodology and innovations introduced by the authors. They propose a system that can interface with data frames through a universal schema, which allows the LLM to answer complex queries

without direct access to the data, thereby preserving privacy. The system translates natural language questions into executable code to perform data operations, with the aim of delivering accurate responses without requiring the LLM to learn from sensitive data directly. The approach is particularly relevant in contexts where data privacy is paramount and can guide the development of similar privacy-preserving tools in dynamic data analytics.

LLM-in-the-loop: Leveraging Large Language Model for Thematic Analysis presents a framework that integrates Large Language Models (LLMs) into the thematic analysis process, enhancing efficiency and potentially transforming qualitative data analysis [15]. By using LLMs in tandem with human coders, the proposed framework optimizes the initial coding and codebook generation phases, demonstrating almost perfect agreement in coding tasks. This innovation significantly reduces the labor intensive aspect of qualitative analysis. The methodology is tested in different case studies, showcasing its adaptability and effectiveness in providing reliable high-quality thematic analysis while addressing the limitation of LLM input size using partial data for codebook development. The paper suggests this collaborative approach could reshape the thematic analysis process, making it more efficient without compromising quality.

A Study on the Implementation of Generative AI Services Using an Enterprise Data-Based LLM Application Architecture utilizes LangChain to orchestrate and integrate various components of a Generative AI system [16]. LangChain is an open-source framework that allows chaining of LLM prompts with external data executions. Within the proposed architecture, LangChain enables translation, summarization, question-answering, text generation, and natural language inference services by chaining different steps. The capability of this tool to integrate with vector databases, such as Chroma or FAISS, and its compatibility with OpenAI's models and GPT4All, showcase a flexible, modular approach to building robust AI services. The framework addresses the need for dynamic, context-aware and efficient use of enterprise data, potentially guiding the development of services where data privacy, security, and accessibility are of utmost importance.

Prompt Sapper: A LLM-Empowered Production Tool for Building AI Chains introduces Prompt Sapper, a tool that enhances the development of AI chains by leveraging large language models [17]. Focusing on the use of LangChain, the paper describes how Prompt Sapper utilizes this library to facilitate the construction of complex AI workflows through a visual programming interface. LangChain plays a pivotal role in enabling a seamless connection between AI prompts and actions, thereby allowing developers to build sophisticated AI-driven applications without the need for extensive coding knowledge. The tool's visual interface simplifies the process of designing, testing, and deploying AI services, aligning with the growing need for accessible AI development platforms that can cater to both technical and nontechnical users.

Other related work focuses on domain-specific applications, for instance, the healthcare domain. Knowledge-Infused LLM-Powered Conversational Health Agent: A Case Study for Diabetes Patients presents an LLM-based Conversational Health Agent (CHA) that incorporates external knowledge, such as dietary guidelines and nutritional information, to provide personalized dietary recommendations for diabetes patients [18]. It demonstrates improved performance over GPT-4 in handling diabetes-related queries by leveraging domain-specific knowledge. The study highlights the CHA's potential to enhance diabetes management through tailored advice but suggests further refinement in integrating diverse data sources and improving user interaction for broader applicability. Further research in this direction with more complex knowledge integration is introduced in [19].

From these recent research outputs, we can see the shift from traditional semantic web technologies like OWL-S and RESTful service description ontologies, to AI-driven approaches such as Large Language Models (LLMs), marks a significant development in this domain. LLMs offer a promise of simplifying complex service and data coordination tasks, highlighted by the inception of tools such as LangChain in the creation of Generative AI services. However, current frameworks such as Prompt Sapper reveal limitations in AI chain development, necessitating tools that enable less technical users to contribute effectively. Similarly, advances in thematic analysis through LLMs and privacy-preserving data interaction models exhibit the potential for enhanced efficiency and security. In healthcare, frameworks like openCHA demonstrate the utility of LLMs in creating personalized, multilingual health agents, while InsightPilot automates data exploration, emphasizing the need for user-friendly interfaces. These technologies, although promising, still face challenges in terms of user interaction, localization with data security considerations, and seamless data integration. Our work aims to build on these advancements, experimental some solutions in real world scenarios for addressing limitations to enhance the automation and user experience in services computing for business analytics.

3 The LLM-centric and Services Oriented Data Analytics Agent Framework

3.1 The Agent Framework

The framework represented in Figure 1 facilitates the development of a Large Language Model (LLM)-centric and Service-Oriented Data Analytics Agent. It operates on the following components:

- Data Mart: it is a term used in Data Warehouse, we adapted the term to present the data layer. The reason is to clearly build a boundary between data management and our research focus. We assume the dataset we are working with has already followed the Data Mart Management concept or similar principles having datasets separated based on different business domains (e.g. HR domain, transaction domain, or operation domain). This foundation layer consists of diverse data repositories, which can be CSV files, SQL databases, NoSQL databases, and unstructured text files. These are the sources from which the system can extract and process data.
- Data Source and Operational Services with Natural Language Descriptions: These services act as intermediaries, interpreting natural language descriptions to understand the semantics and manipulate data sources for various operations. Here the operations can be EDA services, ETL services and algorithm services. EDA and ETL services execute a common data operation based data type but algorithm services are developed to do a specific data analytics task for a certain data. But we call all types of this kind of services as

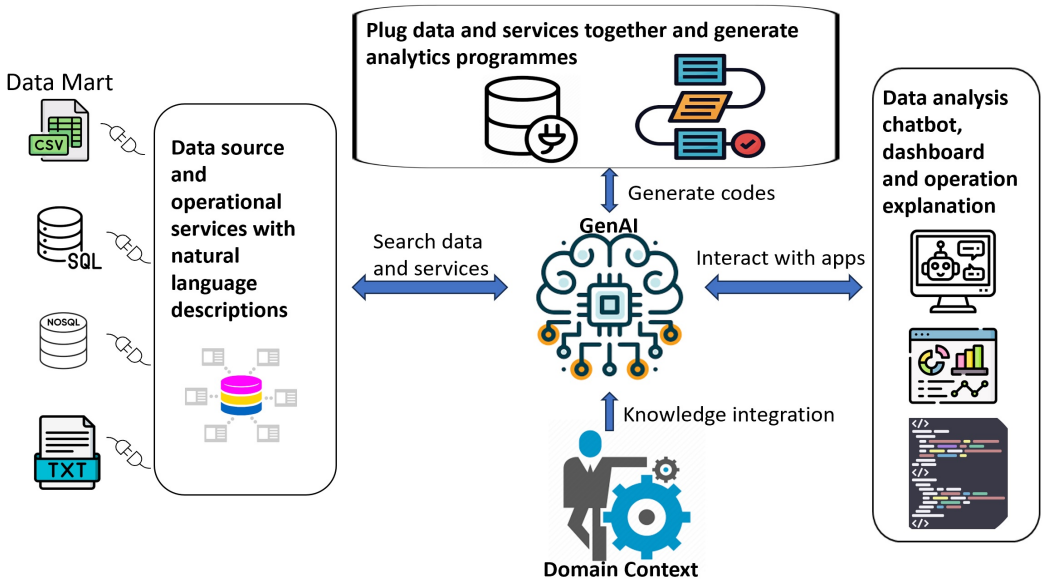


Figure 1: Framework Architecture

MAS because they all support the data analytics task. All this information is indexed in an NLP-enabled simple indexing schema and we will introduce later.

- GenAI Layer: At the core is Generative AI, which performs a multitude of tasks such as generating workflow code, searching data and services, and integrating domain-specific knowledge. Most importantly, the GenAI layer will collect the required data and plug the data into a proper MAS or a sequence of MASs to complete the analysis task and provide the answer to the chatbot interaction. It acts as the brain of the operation, driven by AI to automate analytics and insights extraction.
- Domain Context and Knowledge Integration: This component ensures that the AI's actions are contextualized and informed, using knowledge integration to align with specific domain requirements and enhance the relevance and accuracy of the analytics. The domain context will be added to the prompts in the back-end.
- User Interaction: The system includes interfaces for interaction, such as chatbots for conversational engagement, dashboards for visual representation of data, and explanations for operations, allowing users to easily understand and engage with the system outputs.

The framework demonstrates a dynamic and context-aware solution that goes beyond static service composition, addressing the limitations highlighted in traditional semantic web services. It aims to provide accessible, efficient, and automated data analytics by leveraging the advanced capabilities of LLMs in understanding and generating human-like discourse for complex data interactions. We will demonstrate the workflow of the framework in Case Study and Preliminary Evaluation Results section 4.

3.2 The Micro Analytics Services

In this section, we introduce the concept of Micro Analytics Service (MAS), defined as a specialized function capable of accepting a single input parameter and yielding a single output parameter, specifically within the domain of data analytics. To address a broad spectrum of analytical needs, we have categorized MAS into five distinct service types:

- **Data connector** services are designed to establish connections with specific data repositories or sources, whether located on local or cloud-based systems. Their primary function is to transform the data into a format suitable for analysis, such as converting them into a Python Panda dataframe.
- **Data pre-processing** services focus on preparing data for analysis by implementing methods to clean and preprocess data sources, ensuring they are in an optimal state for further analytical processing.
- **Algorithm** services offer analytical algorithms that interface with data selected or integrated by the GenAI system, facilitating detailed data analysis.
- **Visualisation** services aim to present the results of data analysis in an accessible and comprehensible visual format, enhancing the interpretability of the findings.
- **GenAI tool** services provide functionality for connecting with large language models (LLM) to summarize the results of analytics. Typically, these services use a specific LLM framework, such as OpenAI and LangChain, to process and interpret data.

These services are described using straightforward, human-readable language to accurately convey their functionalities. This semantic clarity enables the GenAI layer to effectively search for and orchestrate the necessary services to respond to inquiries made through chatbot interfaces. An illustrative example of service descriptions

within our framework, particularly for the 'Data connector' category, demonstrates how these services facilitate connections to CSV files and SQLite databases, as shown in Figure 2. The description is a human-language text with simple structure.

```
tool_id = 1
tool_name = pandaloadcsv
tool_description = this tool is to load csv file to panda dataframe data type
tool_location = algorithms/pandaloadcsv.py
tool_function = pandaloadcsv(csv_file)
input_parameter = csv_file
output_parameter = panda_data
category = Data_connector
tool_id = 2
tool_name = connect_sqlite
tool_description = this tool is to connect sqlite database stored in 'sqlite r
tool_location = algorithms/connect_sqlite.py
tool_function = build_sqlite_connector (sqlite_data)
input_parameter = sqlite_database_connecting_string
output_parameter = sqlite_connector
category = Data_connector
```

Figure 2: Service description example

3.3 LangChain and OpenAI prompts Engineering

The integration of LangChain [20] with OpenAI's GPT models represents a cornerstone of GenAI layer in our framework, enabling the understanding of the question for selecting data sources, services, and dynamic generation of analytical workflows through the power of code generation of LLM. This subsection delves into the engineering of prompts that facilitate seamless interaction between the user's natural language queries and the underlying data analytics services.

Prompt engineering is a critical step in harnessing the full potential of LLMs for automated task execution. It involves crafting queries that guide the model towards generating precise and relevant responses. Within our framework, prompts are designed to encapsulate user queries, interpret them in the context of available data and analytics services, and formulate instructions that LangChain can execute to derive the orchestration of the services together.

The LangChain framework represents an innovative approach to integrating Large Language Models (LLMs) like GPT from OpenAI with application-specific logic and external data sources to create more intelligent and context-aware applications. At its core, LangChain facilitates the seamless chaining of language model prompts with actions, allowing developers to construct sophisticated workflows that combine the generative capabilities of LLMs with the precision and reliability of structured data processing and API calls. This bridging enables the creation of advanced AI-driven applications that can understand input from natural language, perform complex data analyses, and generate coherent and insightful responses. By abstracting the complexity involved in interfacing LLMs with diverse data sources and services, LangChain empowers developers to craft intuitive, conversational interfaces for a wide range of applications, from automated data analytics to interactive chatbots, enhancing the accessibility and efficiency of AI technologies across various domains.

Figure 3 provides an illustrative example of a prompt specifically designed to generate Python workflow codes after data selection. Through an iterative process of refinement, encompassing eight rounds of prompt engineering, we achieved a high level of success in the generation and accuracy of these Python workflow codes. This meticulous approach to prompt re-engineering was consistently applied across various aspects of our framework, including data selection, the creation of visualizations, selection of visualization graphs, and the formulation of responses to queries. This iterative refinement process ensures that our system not only accurately interprets user queries, but also effectively translates these queries into actionable insights through a seamless and automated workflow.

```
def codeconnecting(filePath,task_des):
    print('file_path: ', filePath)
    file_extension=get_type_data(filePath)

    with open('algorithms/toolstore.txt') as f:
        lines = f.readlines()
    #print(lines)

    prompt = PromptTemplate(
        input_variables=["file_extension", "task_description", "tool_store"],
        template="Given a file with an extension {file_extension} and a task described as '{task_description}', "
                 "select the best tools from {tool_store} that belong to a tool group that match the {file_extension} "
                 "can be chained together to perform this task. Be aware that the tool_id has no impact on "
                 "the squence of the tools of you selected for the chain. "
                 "Your squence of selected tool chain must correct based on the {file_extension} "
                 "type as you always need to extract the data in first. "
                 "Consider the compatibility of input and output parameters of the tools. "
                 "In the chain, the input and last tool's output must matched. "
                 "Again, produced the best chain of the tools to complete the task. "
                 "Format your answer as a Python dictionary, including tool .py file location, tool names, "
                 "functions should be used, input and output and their sequence of use. "
    )


    print('prompt: ',prompt)

    chain = LLMChain(llm=llm, prompt=prompt)

    # Run the chain only specifying the input variable.
    step1= chain.run(file_extension=file_extension,task_description = task_des,tool_store=lines)
```

Figure 3: Prompt for LangChain Workflow code generation

4 Case Studies

We implemented two case studies to evaluate the proposed framework in terms of the quality of the answers and the accuracy of the data and services connections.

4.1 Case study 1: demographic data analysis

The first implementation is a demographic data analysis case study, the data organised to show demographic information about groups of people according to certain attributes such as age, sex, religion, economic activities, criminal records, location information and place of residence. Furthermore, this dataset contains reported disease cases for each local authorization from 2019 to 2022. In total, there is 111 features in the dataset. This information is very crucial for supporting local government to make policy decisions. The data is CSV file data and there are specific services are implemented in the framework to support data analytics tasks and providing the answering result to the questions.

Figure 4, we illustrate a scenario where a user poses a query about a specific location and year. The GenAI layer acts as an intelligent intermediary, processing the query along with associated details to identify the most relevant data and services. These components are then dynamically linked to conduct the necessary data analytics, including the generation of visualizations, to address the user's inquiry. Subsequently, a well-coordinated series of services,

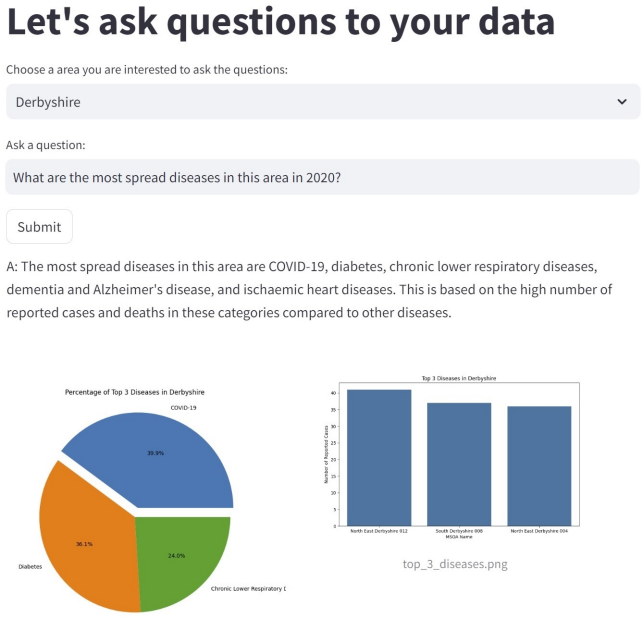


Figure 4: Case Study 1: Demographic data analysis

or a workflow, is orchestrated to execute the analytics task. Upon completion, the results are relayed back to the GenAI layer, which then employs these insights to respond to the query and determines the most suitable visualizations for display in the Chatbot interface. Figure 5 presents the backend orchestration workflow code, crafted in real-time by the GenAI layer. This code is generated based on service and data descriptions provided in a straightforward text file, utilizing plain English without the need for complex semantic frameworks, thereby streamlining the process of turning user inquiries into actionable analytics.

```
# Import necessary modules from the algorithms directory
from algorithms.connectors.connect_csv import pandaloadcsv
from algorithms.data_preprocessing import clean
from algorithms.region_analysis_report import location_summarisation
from algorithms.mapviz import createMapViz

# Define input file path
input_file = 'data\Sum.csv'

# Invoke the first tool, pandaloadcsv, passing in the input file as
panda_data = pandaloadcsv(input_file)

# Invoke the second tool, clean, passing in the output of the first
clean_panda_data = clean(panda_data)

# Invoke the third tool, location_summarisation, passing in the outp
region_analysis_answer = location_summarisation(clean_panda_data)

# Invoke the fourth tool, createMapViz, passing in the output of the
map_visualisation = createMapViz(clean_panda_data)
```

Figure 5: Workflow generated by LangChain with OpenAI APIs for case study one

4.2 Case study 2: business data analysis

In our second case study, which focuses on business data analysis, we leveraged open source data to emulate the operations of a company using two distinct datasets. The first dataset is an HR dataset that covers employment-related information which is stored in a SQL database. The second dataset is the CSV data provides insights into product sales across various local branch shops. This case study highlights the challenge of dynamically orchestrating data analytics to address diverse queries—ranging from employment to sales-related issues and even comparisons across different branches of the business.

Figure 6 shows the dynamic workflow code that was generated in response to a query on the number of employees over the age of 50 in the HR database. This database is managed using an SQLite service, demonstrating the flexibility of our system to adapt and respond to specific analytical queries by generating targeted workflow codes. In addition, the service can communicate with LLM to generate SQL query on the fly to get the answer. Through this example, we illustrate the system's capability to handle complex, varied data sources and produce relevant, actionable insights tailored to specific business questions.

```
# Import necessary modules
from algorithms.connectors.connect_sqlite import build_sqlite_connector
from algorithms.query_sqlite import query_sqlite_database

# Define input parameter for first tool
sqlite_database_endpoint = 'data/hr_data/hr_database.sqlite'

# Invoke first tool to build SQLite connector
conn = build_sqlite_connector(sqlite_database_endpoint)

# Invoke second tool to query SQLite database using the connector from
string_type_of_answer_from_sql = query_sqlite_database(conn)

# Print the output of the second tool
print(string_type_of_answer_from_sql)
SELECT COUNT(*) FROM hr_data WHERE Age > 50;
There are 143 employees who are older than 50 years old. This is determ
 result. This function is used to return the number of rows that match
# Import necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Read the data from the existing dataframe
df = pd.DataFrame({'COUNT(*)': [143]})

# Set the seaborn style to "darkgrid"
sns.set_style("darkgrid")

# Create a bar plot using seaborn
sns.barplot(x=df.index, y="COUNT(*)", data=df)

# Set the title of the plot
plt.title("Count of Stored Items")

# Save the plot as a png file in the "vis" folder
plt.savefig("vis/Count_of_Stored_Items.png")

# Create a scatter plot using seaborn
sns.scatterplot(x=df.index, y="COUNT(*)", data=df)
```

Figure 6: Workflow generated by LangChain with OpenAI APIs for case study two

5 Preliminary Evaluations

To assess the efficacy and reliability of our proposed framework, we designed an evaluation strategy focused on five key metrics:

Data Connection Accuracy, Code Generation Success, Code Output Accuracy, Visualization Code Generation Success, and Visualization Understandability. These metrics were chosen to cover the critical aspects of our framework's functionality, from data retrieval and processing to the generation of code and visualizations. The evaluation results are shown on the Tabel 1.

5.1 Evaluation Metrics

Data Connection Accuracy: Measures the success rate of correctly establishing connections to data sources. We applied 20 unit test cases.

Code Generation Success: Assesses the ability of our system to generate executable code based on the chatbot inputs. This test only if the code is runnable or not, without considering if the output of the code is correct or not. We uses the same 20 examples in the data connection accuracy test cases.

Code Output Accuracy: Evaluates the correctness of the code output against expected results, crucial for ensuring reliable data analysis. We also applied 20 unit test cases.

Visualization Code Generation Success (Vis Code Generation Success): Determines the success rate of generating code for data visualizations, and the code can be executed to show the visualized graphics.

Visualization understandability (Vis understandable): Quantifies the ease with which the generated visualizations is suitable to be understood by users, indicating the effectiveness of data visualization.

5.2 Methodology

Our evaluation involved interacting with the framework through a chatbot interface by posing 20 designed diverse questions for each case study that span the functionalities covered by our evaluation metrics. This approach simulates real-world usage scenarios, providing insights into the framework's performance in practical applications. The first three metrics—Data Connection Accuracy, Code Generation Success, and Code Output Accuracy—were measured using Python unit testing to ensure precise and objective evaluation.

5.3 Results

The evaluation results, summarized in Table 1, reveal varying performance in the two case studies. Case Study 1 demonstrated perfect Data Connection Accuracy (100%), while Case Study 2 scored lower at 90%. Code Generation Success and Code Output Accuracy also varied, with Case Study 1 outperforming Case Study 2 (95% vs. 100% and 95% vs. 100%, respectively). Visualization Code Generation Success was notably higher in Case Study 2 (95%) compared to Case Study 1 (70%). However, both case studies struggled with Visualization Understandability, scoring 50% and 80%, respectively.

5.4 Discussion

The evaluation highlights strengths in data connection and code generation, particularly in Case Study 1. The lower Data Connection Accuracy in Case Study 2 (90%) was caused by the different types and domains of the data sources involved (SQL and CSV), suggesting the need for enhanced robustness in data source selection requires engineering.

Table 1: Evaluation Results for Case Studies

Metric	Case Study 1 (%)	Case Study 2 (%)
Data Accuracy	100	90
Code Success	95	85
Output Accuracy	100	100
Vis Code Success	70	95
Vis understandable	50	80

The significant discrepancies in Code Generation Success and Code Output Accuracy between the case studies, with Case Study 2 scoring notably lower, indicate challenges in handling the complexity of queries and data involved. These findings underscore the importance of improving our code generation algorithms, especially for more complex analysis tasks.

However, the correctness rates of 90% and 85% in the first two matrices are still very positive signs that we need to continue working in this direction.

In addition, We also observed that once the code can be successfully generated, the output accuracy (the answers to the analytics questions) will be 100%.

The Visualization Code Generation Success metric presents an interesting contrast, with Case Study 2 excelling. This suggests that while our framework can effectively generate visualization code for complex datasets, the understandability of these visualizations remains a challenge, as the major problems of providing meaningful labels and titles on the visualization graphs. This discrepancy underscores the need for a focus on generating not only accurate but also clear and interpretable visualizations.

The notably low rate of Visualization Understandability in Case Study 1 (50%) highlights a critical area for improvement. Ensuring that visualizations are not only generated successfully but are also intuitive and informative for users is paramount. This may involve integrating more sophisticated design principles into the visualization generation process or incorporating user feedback loops to refine output. However, we observe that SQL-based queries can perform better in visualization code generation and understandability.

Our evaluation has demonstrated the potential of our framework in automating data analytics tasks through a chatbot interface, with strong performance in data connection accuracy and code generation. However, the results also reveal areas for improvement, particularly in the understandability of generated visualizations and the handling of complex data analysis tasks. Future work will focus on addressing these challenges to enhance the framework's usability and reliability further.

6 Conclusion

In our exploration, we have integrated Large Language Models (LLMs) with service-oriented computing, inaugurating a pioneering framework that champions a communicable, dynamic approach to data analytics. This synthesis is not merely a technical feat; it represents a paradigm shift in how data analytics can be approached, making complex analyses accessible through conversational interfaces. Our framework stands out by transforming the intricate

landscape of data analytics into an intuitive, dialogue-driven exploration, significantly lowering the barrier to sophisticated data interaction and interpretation.

The case studies underscore the framework's adeptness in seamlessly connecting to data sources, generating and executing code, and rendering insightful visualizations through natural language prompts. While showcasing robust performance, these evaluations also candidly reveal areas for enhancement, particularly in making the generated visualizations more interpretable and refining the handling of intricate data analysis challenges. Such insights are crucial, delineating a roadmap for augmenting the framework to meet the nuanced demands of diverse data environments.

Crucially, the integration of LLMs with service-oriented computing heralds a transformative tool for Small and Medium-sized Enterprises (SMEs), democratizing access to advanced analytics. By abstracting the complexities of programming and data management, our framework empowers SMEs to harness powerful business insights through straightforward conversational queries. This capability enables businesses to focus on strategic decision-making rather than the intricacies of data science, potentially revolutionizing how SMEs engage with their data and analytics, driving informed decisions and fostering innovation.

As we advance, our focus will be on refining the visualization aspects and enhancing the system's adeptness at navigating and analyzing complex data sets. By addressing these focal points, we aim to elevate the user experience, ensuring that our framework not only serves as a bridge between sophisticated data analytics and end-users but also becomes a cornerstone in the toolkit of businesses, especially SMEs, seeking to leverage data-driven insights without the traditional barriers of high technical expertise. In addition, we will work on evaluations of different LLMs.

References

[1] T. Chugh, K. Tyagi, R. Seth and P. Srinivasan, "Intelligent agents driven data analytics using Large Language Models," 2023 International Conference on Artificial Intelligence, Blockchain, Cloud Computing, and Data Analytics (ICoABCD), Denpasar, Indonesia, 2023, pp. 152-157, doi: 10.1109/ICoABCD59879.2023.10390973.

[2] A. Alford, *OpenAI Announces GPT-3 AI Language Model with 175 Billion Parameters*, InfoQ, 2020. [Online]. Available: https://www.infoq.com/news/2020/06/openai-gpt3-model/

[3] *What's the next word in large language models?*, Nature Machine Intelligence, 2023. [Online]. Available: https://www.nature.com/articles/s42256-023-00655-z

[4] D. Martin et al., *OWL-S: Semantic Markup for Web Services*, W3C Member Submission, 2004. [Online]. Available: https://www.w3.org/submissions/OWL-S/

[5] "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language," W3C Recommendation, June 2007. [Online]. Available: https://www.w3.org/TR/wsdl

[6] *Science in the age of large language models*, Nature Reviews Physics, 26 April 2023. [Online]. Available: https://www.nature.com/articles/s42254-023-00581-4

[7] *A Comprehensive Overview of Large Language Models*, ar5iv.org, 2023. [Online]. Available: https://arxiv.org/abs/2307.06435

[8] *LLMRec: Large Language Models with Graph Augmentation for Recommendation*, ar5iv.org, 2023. [Online]. Available: https://arxiv.org/abs/2311.00423

[9] Hogan, Aidan. *The Semantic Web: Two Decades On*, Semantic Web 0 (0) 1 1, IOS Press, 2020. Available: https://aidanhogan.com/docs/semantic-web-now.pdf, last visit: [7th March 2024]

[10] Yu HQ, O'Neill S, Kermanizadeh A. *AIMS: An Automatic Semantic Machine Learning Microservice Framework to Support Biomedical and Bioengineering Research*. Bioengineering (Basel). 2023 Sep 27;10(10):1134. doi: 10.3390/bioengineering10101134. PMID: 37892864; PMCID: PMC10603862.

[11] PandasAI, Available: https://docs.pandas-ai.com/en/latest/

[12] M. Aiello and I. Georgievski, "Service composition in the ChatGPT era," SOCA, vol. 17, pp. 233–238, 2023. [Online]. Available: https://doi.org/10.1007/s11761-023-00367-7

[13] S.-C. Liu et al., "JarviX: A LLM No-code Platform for Tabular Data Analysis and Optimization," in Proc. of the 2023 Conf. on Empirical Methods in Natural Language Processing: Industry Track, Singapore, 2023, pp. 622–630.

[14] J. Ye, M. Du, and G. Wang, "DataFrame QA: A Universal LLM Framework on DataFrame Question Answering Without Data Exposure," arXiv preprint arXiv:2401.15463, 2024.

[15] S.-C. Dai, A. Xiong, and L.-W. Ku, "LLM-in-the-loop: Leveraging Large Language Model for Thematic Analysis," in Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 9993–10001, Singapore, 2023.

[16] C. Jeong, "A Study on the Implementation of Generative AI Services Using an Enterprise Data-Based LLM Application Architecture," Adv. in Artif. Intell. and Mach. Learn., vol. 03, no. 04, pp. 1588–1618, 2023. [Online]. Available: http://dx.doi.org/10.54364/aaiml.2023.1191

[17] Y. Cheng, J. Chen, Q. Huang, Z. Xing, X. Xu, and Q. Lu, "Prompt Sapper: A LLM-Empowered Production Tool for Building AI Chains," arXiv preprint arXiv:2306.12028, 2023.

[18] M. Abbasian, Z. Yang, E. Khatibi, P. Zhang, N. Nagesh, I. Azimi, R. Jain, and A. M. Rahmani, "Knowledge-Infused LLM-Powered Conversational Health Agent: A Case Study for Diabetes Patients," arXiv preprint arXiv:2402.10153, 2024.

[19] M. Abbasian, I. Azimi, A. M. Rahmani, and R. Jain, "Conversational Health Agents: A Personalized LLM-Powered Agent Framework," arXiv preprint arXiv:2310.02374, 2024.

[20] LangChain, https://www.langchain.com/