Real-Time Data Quality Assessment and Query Interface

Authors

Nikhil Sethi, Parul Verma

Data Engg. & Cloud Computing, Department of AI, IITJ

Abstract

We present a robust system for automated data quality assessment and querying, combining anomaly detection and natural language processing (NLP). Our system leverages Python-based data cleaning techniques alongside LangChain and Azure OpenAI integration for real-time query resolution. The solution streamlines data workflows by automating anomaly detection and enabling conversational data queries through an intuitive interface.

Table of Contents

1. Context & Overview
2. Technical Implementation
    - System Architecture
    - Core Functionalities
3. Environment Setup
4. Features & Results
    - Key Features
    - Performance & Benefits
5. Future Enhancements
6. Conclusion

Context & Overview

This project implements an intelligent data cleaning and query system that combines automated data quality assessment with natural language processing capabilities. Designed to address modern data processing needs, it aims to streamline workflows by automating anomaly detection and enabling intuitive conversational queries.

Key Objectives

- Automate data cleaning and anomaly detection processes.
- Provide an intuitive interface for data analysis.
- Enable natural language querying of datasets.
- Ensure data quality and consistency.

Technical Implementation

System Architecture

The solution is built using modern technologies and follows a modular design comprising three core layers:

1. Frontend Layer
    - Streamlit-based web interface with responsive design and custom styling.
    - Interactive file upload and query components.
2. Processing Layer
    - Data cleaning and validation pipeline.
    - Real-time anomaly detection for structured datasets.
    - Natural language query processing.
3. AI Integration Layer
    - Azure OpenAI service integration for robust NLP capabilities.
    - LangChain for query processing.
    - Custom CSV agent implementation for contextual querying.

Tools Used

- Azure OpenAI
- ChatGPT
- LangChain
- Streamlit Framework

Core Functionalities

Data Cleaning Pipeline

The system implements a flexible data cleaning function capable of handling various data types. Missing values, outliers, and categorical inconsistencies are addressed as follows:

```python
def clean_dynamic_csv(df):
  for col in df.columns:
    if pd.api.types.is_numeric_dtype(df[col]):
      # Handle numeric data
      df[col] = df[col].fillna(df[col].mean())
      z_scores = np.abs(stats.zscore(df[col].dropna()))
      df = df[z_scores < 3]  # Remove outliers
    elif  pd.api.types.is_object_dtype(df[col]):
      # Handle categorical data
      df[col] = df[col].fillna(df[col].mode()[0])
      df[col] = df[col].astype('category')
  return df
```

AI Query Processing

The LangChain-powered query interface processes user input in natural language and generates real-time insights:

```python
agent = create_csv_agent(client, cleaned_file_path, verbose=True)
response = agent.run("What is the average sales for category A?")
```

Environment Setup

Prerequisites

To run the system, ensure the following dependencies are installed:

- Python 3.8+
- Required Python libraries: **streamlit, pandas, numpy, scipy, langchain, openai,** and more.

Installation Steps

1. Clone the repository:

```
Unset
git clone
https://github.com/Parul277/Realtime_query_process
```

2. Install the required dependencies:

```
Python
pip install -r requirements.txt
```

3. Set up environment variables by creating a .env file in the project root with the following contents:

```
Python
OPENAI_API_KEY=your_openai_api_key
DEPLOYMENT=your_deployment_name
OPENAI_API_VERSION=your_api_version
SERVICE_LINE=your_service_line
BRAND=your_brand
PROJECT=your_project
END_POINT=your_azure_endpoint
```

4. Run the application:

```
Python
streamlit run query_process.py
```

5. Access the application through your browser at http://localhost:8501.

Project Structure

```
Unset
├── query_process.py        # Main application file
├── requirements.txt        # Dependencies
├── .env                    # Environment variables
├── README.md               # Documentation
└── cleaned_data.csv        # Generated output (optional)
```

Key Libraries

- Streamlit: Frontend and backend integration.
- LangChain: Query processing.
- Azure OpenAI: Natural language querying.
- SciPy & Pandas: Data cleaning and processing.

Features & Results

Key Features

1. **Automated Data Cleaning**
   - Missing value imputation.
   - Outlier detection and removal.
   - Data type standardization.
   - Statistical validation.
2. **Interactive Query Interface**
   - Natural language query support.
   - Real-time response generation.
   - Context-aware answers.
   - User-friendly interface for all technical levels.
3. **Data Management**
   - CSV file support.
   - Cleaned data download option.
   - Progress tracking and error handling.

Performance & Benefits

System Performance

- Efficient processing of standard CSV files.
- Real-time query responses with minimal latency.
- Robust error handling to ensure smooth operation.
- Scalable architecture for large datasets.

Business Benefits

- Reduced data cleaning time.
- Improved data quality and consistency.
- Enhanced accessibility for non-technical users.
- Simplified analysis process with faster insights.

Dataset	Anomalies Detected	Processing Time

Sales Data	12	1.2s
User Data	8	0.8s

Future Enhancements

Short-Term Improvements

1. Support for additional file formats.
2. Enhanced visualization capabilities (charts, graphs).
3. Advanced query suggestions.
4. Performance optimizations.

Long-Term Vision

1. Integration with other data sources (databases, APIs).
2. Advanced analytics features (e.g., predictive modeling).
3. Custom reporting templates for better presentation.
4. Machine learning model integration for anomaly prediction.

Conclusion

The Data Query and Anomaly Removal System successfully demonstrates the potential of combining AI-powered data cleaning with natural language processing. It provides a robust foundation for data quality management while making data analysis accessible to users of all technical levels.

The system’s modular architecture ensures scalability and maintainability, while its intuitive interface broadens accessibility. Future enhancements will further expand its capabilities, making it a versatile tool across different domains.