Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2021, 8(10):96-99

Research Article ISSN: 2394 - 658X

Efficiently Handling Streaming JSON Data: A Novel Library for GCS-to-BigQuery Ingestion

Preyaa Atri

Preyaa.atri91@gmail.com

ABSTRACT

This paper examines a Python library designed to efficiently stream JSON data from Google Cloud Storage (GCS) to BigQuery tables. The library offers functionalities for handling large datasets, enriching data with timestamps and filenames, and ensuring compatibility with BigQuery's schema requirements. Additionally, the library demonstrates potential use cases not only in the realm of data engineering but also in advancing AI development by facilitating robust data ingestion pipelines. We analyze the library's features, potential use cases, and its impact on both data engineering workflows and AI model training. Furthermore, we discuss its limitations and propose recommendations for further development.

Keywords: BigQuery, Google Cloud Storage, JSON, Data Streaming, Data Enrichment, AI

INTRODUCTION

BigQuery, a managed data warehouse service by Google Cloud Platform (GCP), plays a crucial role in modern data pipelines. Efficiently ingesting data into BigQuery from various sources is essential for large-scale data analytics and AI model training. This paper explores a Python library that streamlines the process of loading JSON data files stored in GCS buckets into BigQuery tables. This library not only addresses data engineering challenges but also provides significant advancements for AI development by ensuring the availability of clean, enriched data for training robust models.

Efficient data ingestion is vital for both data engineering and AI development. As AI models require large volumes of data for training and validation (Roh et al., 2021), the seamless integration of this library into data pipelines ensures that data scientists and engineers can focus on model development rather than data handling. The ability to process data in chunks and enrich it with metadata enhances the quality and traceability of datasets, which is critical for maintaining high standards in AI research and development.

PROBLEM STATEMENT

Loading large JSON datasets into BigQuery can be challenging due to memory limitations and schema compatibility issues. Traditional approaches might involve loading the entire dataset at once, leading to memory constraints. Additionally, special characters in JSON field names can cause compatibility problems with BigQuery's schema definition.

SOLUTION

The GCS to BigQuery Streaming JSON Processor library addresses these challenges by providing a chunked processing mechanism. The library reads and processes JSON data in smaller chunks (configurable via the chunk_size parameter), optimizing memory usage and enabling efficient handling of large datasets. This approach aligns with research by (Mustafa et al., 2014) who emphasize the importance of chunk-based processing for handling massive data streams.

FUNCTIONALITY

The GCS to BigQuery Streaming JSON Processor library provides functionalities to efficiently and seamlessly transfer JSON data from GCS to BigQuery. Here's a breakdown of its key features:

- **Chunk-wise Processing:** This core functionality enables processing JSON data in manageable chunks as defined by the chunk_size parameter. This optimizes memory usage and facilitates handling large datasets by dividing the data into smaller portions for BigQuery load jobs.
- **Data Enrichment (Optional):** The library offers the ability to augment the data with additional columns during processing. Users can enable adding a load_date column containing the current timestamp for each record. Additionally, they can choose to include a source_file_date column reflecting the creation timestamp of the processed JSON file, and a filename column containing the original filename of the JSON file. This data enrichment enhances data lineage and simplifies downstream analysis, ensuring trust and understanding in big data systems.
- **Improved BigQuery Compatibility:** The library ensures seamless integration with BigQuery's schema requirements by automatically cleaning column names. It removes special characters from the JSON field names, preventing potential errors during data loading into BigQuery tables.

INSTALLATION INSTRUCTIONS

The library can be easily installed using the pip package manager. Here's the command to install the required libraries:

Bash

```
pip install google-cloud-storage google-cloud-bigquery pandas #installs dependencies

pip install JSON_file_streaming_GCS_BigQuery #installs GCS to BigQuery Streaming
JSON Processor library
```

These libraries provide the necessary functionalities for interacting with GCS, BigQuery, and data manipulation within the streaming process.

USAGE

```
Python
from JSON_file_streaming_GCS_BigQuery import process_json_file_streaming

# Replace placeholders with your specific values
dataset_id = "your_dataset_id"
table_name = "your_table_name"
project_id = "your_project_id"
bucket_name = "your_bucket_name"
source_folder_name = "your_source_folder"
destination_folder_name = "your_destination_folder" # Optional

# Optional arguments (default values provided)
chunk_size = 10000 # Adjust chunk size for memory management
add_record_entry_time = True
add_source_file_creation_time = False
add_file_name = True

process_json_file_streaming(dataset_id, table_name, project_id,
 source_folder_name, destination_folder_name,
 chunk_size, add_record_entry_time,
 add_source_file_creation_time, add_file_name)
```

The library offers a user-friendly interface with clear parameters for configuration. Here's an example demonstrating its usage:

In this example, we define the BigQuery dataset ID, table name, project ID, and bucket names containing the JSON data. We can optionally specify a destination folder within the bucket to move the processed JSON files. Additionally, we can adjust the chunk size for memory management and enable/disable data enrichment options. Finally, the process_json_file_streaming function is called with the specified arguments to initiate the streaming process.

DEPENDENCIES AND CONSIDERATIONS

Authentication: It's crucial to have proper authentication set up for accessing GCS and BigQuery. Users need to configure their GCP environment with appropriate credentials before utilizing the library.

Data Validation: The library currently assumes that the JSON files stored in GCS are well-formatted and valid. Implementing data validation checks within the library would enhance robustness by ensuring data integrity before loading into BigQuery.

Error Handling: The current error handling mechanisms might require further customization for production environments. More granular control over failures can be achieved by incorporating logging, retry logic, and notifications for critical issues.

LIMITATIONS AND FUTURE SCOPE

The GCS to BigQuery Streaming JSON Processor library offers valuable functionalities, but it's important to acknowledge its limitations and potential for future development.

Current Limitations:

- **Dependency on User-configured Authentication:** The library assumes that users have already set up proper authentication for accessing GCS and BigQuery. This requires independent configuration within the GCP environment before utilizing the library.
- **Limited Data Validation:** The library currently operates under the assumption that the JSON data files stored in GCS are well-formatted and valid. Implementing data validation checks within the library itself would enhance its robustness by ensuring data integrity before loading into BigQuery. These checks could involve schema validation and identifying missing or invalid values within the JSON files.
- **Basic Error Handling:** The current error handling mechanisms might require further customization, particularly for production environments. More granular control over failure scenarios could be achieved by incorporating features such as logging specific errors, implementing retry logic for transient failures, and generating notifications for critical issues.

Future Scope: Expanding the Library's Capabilities

The GCS to BigQuery Streaming JSON Processor library offers valuable functionalities, but its future potential can be significantly expanded to benefit both data engineering and AI development.

- **Integrated Data Validation:** Incorporating data validation checks directly within the library would streamline the data ingestion process. This could involve schema validation against a user-defined schema or predefined schema templates, along with checks for missing or invalid values within the JSON data. Enhanced data validation is crucial for ensuring the integrity of datasets used in AI training, thereby improving model performance.
- **Advanced Error Handling:** Implementing more sophisticated error handling mechanisms would provide users with finer-grained control over how the library handles failures. This could include logging specific errors with detailed information for troubleshooting, employing retry logic with backoff strategies for transient errors, and generating notifications for critical issues that require immediate attention. Improved error handling would ensure the reliability of data pipelines, which is essential for continuous AI model training and deployment.
- **Schema Evolution Support:** The flexibility of JSON structures allows users to work with data without rigidly defining a schema beforehand, facilitating the management of ad hoc and changing data with evolving schemas (Wang et al., 2015). Exploring options for handling schema evolution would enhance the library's flexibility. This could involve dynamically adapting to changes in the JSON data structure by allowing users to specify schema mapping rules or incorporating automatic schema inference capabilities. These features would make the library more adaptable to evolving data

pipelines, benefiting both data engineering and AI development by providing up-to-date, accurate datasets for model training.

By addressing these limitations and incorporating the suggested future developments, the GCS to BigQuery Streaming JSON Processor library can become an even more robust and versatile tool for data engineers and AI developers working with JSON data pipelines in the GCP environment.

CONCLUSION

The GCS to BigQuery Streaming JSON Processor library provides a valuable tool for data engineers and AI developers working with JSON data in GCP. Its functionalities for chunked processing, data enrichment, and schema cleaning streamline data ingestion workflows and improve BigQuery integration. By addressing the identified limitations through data validation and enhanced error handling, the library's capabilities can be further strengthened, thereby advancing the fields of data engineering and AI development. The integration of this library into data pipelines ensures the availability of high-quality, enriched datasets necessary for training robust AI models, ultimately contributing to the advancement of AI technologies.

REFERENCES

[1]. A. Mustafa, A. Haque, L. Khan, M. Baron, & B. Thuraisingham, "Evolving stream classification using change detection", Proceedings of the 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, 2014. https://doi.org/10.4108/icst.collaboratecom.2014.257769

[2]. L. Wang, S. Zhang, J. Shi, L. Jiao, O. Hassanzadeh, J. Zouet al., "Schema management for document stores", Proceedings of the VLDB Endowment, vol. 8, no. 9, p. 922-933, 2015. https://doi.org/10.14778/2777598.2777601

[3]. Google Cloud Platform. [Online]. Cloud Storage Documentation. Available: https://cloud.google.com/storage/docs

[4]. Google Cloud Platform. [Online]. BigQuery Documentation. Available: https://cloud.google.com/bigquery/docs

[5]. Pandas development team. [Online]. pandas.DataFrame.to_gbq. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_gbq.html

[6]. Y. Roh, G. Heo, & S. Whang, "A survey on data collection for machine learning: a big data - ai integration perspective", IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 4, p. 1328-1347, 2021. https://doi.org/10.1109/tkde.2019.2946162