International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

The Making of an Data Pipeline

Harsh Kaushik[1], Avnish Rai[2], Gaurav Kapasiya[3], Jai Prakash Bhati[4]

[1,2,3]Student, IIMT College of Engineering
[4]Professor, IIMT College of Engineering

Abstract

This paper details the development and implementation of a data engineering pipeline designed for the extraction, transformation, and loading (ETL) of data from a web-based directory. The project involves using asynchronous web scraping techniques to gather user details from a local business directory, transforming the data into a structured format, and loading it into a storage solution. The pipeline utilises Python, the HTTPX library for asynchronous HTTP requests, BeautifulSoup for HTML parsing, and Amazon S3 for data storage. By leveraging these technologies, the pipeline demonstrates an efficient approach to handling large-scale web data extraction and processing, significantly reducing the time required to gather and organise data from multiple web pages. This paper provides insights into the architecture, implementation, and performance of the ETL pipeline, highlighting the benefits and challenges of using asynchronous programming in data engineering.

1. Introduction

In today's data-driven world, the ability to extract, transform, and load data from various sources is crucial for businesses and researchers alike. Data engineering pipelines play a pivotal role in this process, enabling the efficient collection and processing of vast amounts of data. Web scraping,a method for extracting data from websites, is particularly useful for gathering publicly available information from the internet. However, traditional web scraping methods can be time-consuming and resource-intensive, especially when dealing with large datasets or multiple web pages.

This paper presents the development of a robust ETL pipeline designed to scrape user details from www.local.ch, **a local business directory**. The pipeline leverages asynchronous programming techniques to enhance performance and scalability, making it capable of handling a large number of concurrent web requests. The use of HTTPX for asynchronous HTTP requests, BeautifulSoup for HTML parsing, and Amazon S3 for data storage ensures that the pipeline is both efficient and reliable.

The implemented ETL pipeline not only focuses on efficiency but also emphasises data accuracy and integrity. By integrating advanced error-handling mechanisms and retry strategies, the pipeline minimises data loss and ensures the completeness of the extracted information.

2. LITERATURE SURVEY

Data engineering has become an essential discipline in the era of big data, enabling the efficient processing, management, and transformation of vast amounts of data. Data pipelines are fundamental in this context, facilitating the flow of data from various sources to storage and analytical systems. This literature survey explores key contributions and methodologies in data engineering, focusing on notable data pipelines developed by researchers.

Google's Dataflow model

Presented by Akidau et al. (2015), introduced a unified programming model and managed service for batch and stream data processing. It laid the groundwork for the Apache Beam project, which allows developers to create data pipelines that can run on various processing engines, including Apache Flink and Apache Spark.

Apache Hadoop and MapReduce

Dean and Ghemawat's (2008) paper on MapReduce introduced a programming model for processing large datasets with a distributed algorithm on a cluster. This model became the cornerstone of the Apache Hadoop project, which revolutionised big data processing by providing a scalable, fault-tolerant framework..

Lambda Architecture

Nathan Marz (2015) proposed the Lambda Architecture, which is designed to handle massive quantities of data by utilising both batch and stream processing methods. This architecture addresses the need for real-time analytics while ensuring data consistency and scalability.

Kappa Architecture

Jay Kreps (2014) introduced the Kappa Architecture as an alternative to the Lambda Architecture, aiming to simplify the data processing pipeline by using stream processing alone. This approach reduces complexity and latency by avoiding the need for separate batch processing systems.

ETL Pipelines with Apache and NiFI

Apache NiFi, initially developed at the NSA, provides a robust data ingestion and distribution framework. Guhathakurta et al. (2017) highlighted its capabilities in building scalable, reliable ETL pipelines that support data provenance and security.

Data Engineering with Airflow

Apache Airflow, developed by Maxime Beauchemin (2015) at Airbnb, has become a popular open-source tool for orchestrating complex data workflows. It allows users to programmatically author, schedule, and monitor data pipelines.

ETL Pipelines in Cloud Environments

Data pipelines in cloud environments have become increasingly important due to the scalability and flexibility of cloud services. Amazon's AWS Glue, Azure Data Factory, and Google Cloud Dataflow are notable examples. These services simplify the creation, scheduling, and monitoring of ETL workflows, enabling efficient data integration and processing in the cloud.

3. Data Characteristics

In the context of this project, we focus on scraping user details from www.local.ch, a prominent local business directory. The data extracted from this site exhibits several distinct characteristics that are crucial for the subsequent processing stages. Understanding these characteristics ensures the development of an efficient and robust ETL pipeline. Key characteristics of the data are outlined below:

- **User Details**: The primary focus is on extracting detailed user information, including names, addresses, and contact numbers. This data is typically structured within HTML elements that need to be accurately parsed to ensure completeness.
- **Data Volume**: Given the comprehensive nature of www.local.ch, the volume of data can be substantial. This necessitates the use of asynchronous programming to handle numerous concurrent web requests efficiently.

- **Data Variability**: The data may vary significantly in terms of format and completeness. Different business listings might present user details in various ways, necessitating flexible parsing methods.
- **Frequency of Updates**: Business listings on www.local.ch are frequently updated to reflect current information. This characteristic requires the pipeline to be capable of regularly updating the dataset without redundancy.
- **Data Quality Issues**: Common issues include incomplete records, duplicates, and inconsistencies in formatting. These issues necessitate thorough data validation, deduplication, and transformation processes.
- **HTML Structure**: The structure of the HTML pages can vary, and it is essential to develop robust parsing techniques using BeautifulSoup to navigate these variations effectively.

4. Methodology

The methodology for developing the ETL pipeline to scrape user details from www.local.ch involves a structured approach encompassing data collection, preprocessing, transformation, feature selection, and storage. The proposed method is represented in several stages, as detailed below:

A. Data Collection

Data collection is the foundational phase of the ETL pipeline. This involves making asynchronous HTTP requests to www.local.ch, retrieving the HTML content, and parsing it to extract relevant user details. The process is implemented using the following steps:

1. **Setting Up Asynchronous HTTP Requests**: Using the HTTPX library, asynchronous HTTP requests are made to www.local.ch to retrieve HTML pages containing business listings.
2. **Navigating URLs**: The base URL is dynamically constructed to navigate through multiple pages of listings, ensuring comprehensive data collection.
3. **HTML Parsing**: BeautifulSoup is used to parse the HTML content and locate elements containing user details such as names, addresses, and contact numbers.

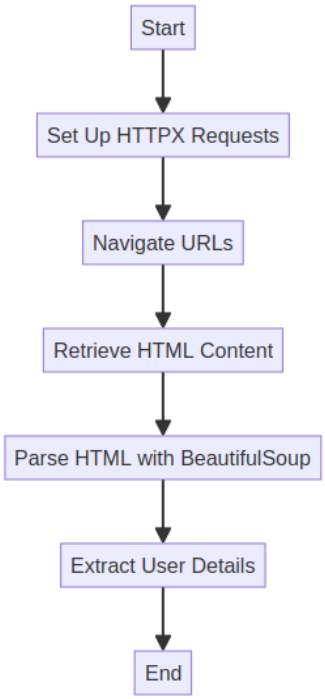


Fig. Data Retrieval

B. Data Preprocessing

Preprocessing ensures that the raw data collected is cleaned and formatted appropriately for further processing. This involves:

1. **Data Validation**: Verifying the presence and correctness of key fields such as phone numbers and addresses using regular expressions and lookup tables.
2. **Deduplication**: Identifying and removing duplicate records to maintain a clean dataset. Techniques like hashing and fuzzy matching are used to detect duplicates.
3. **Error Handling**: Implementing error-handling mechanisms to manage issues like missing fields or malformed data entries.

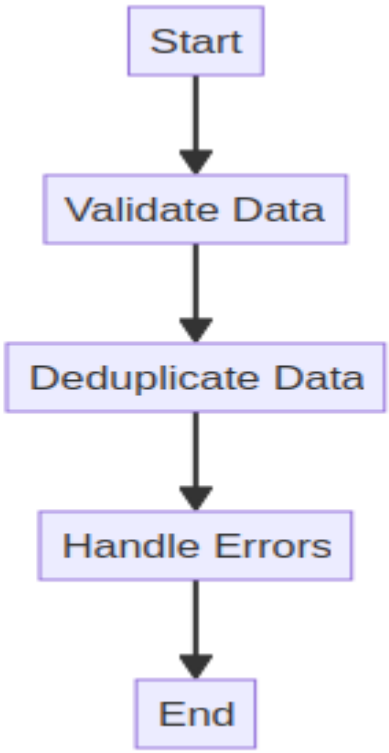


Fig. Data Processing

C. Data Transformation

Data transformation involves converting the data into a format suitable for analysis and storage. This includes:

1. **Standardising Formats**: Converting phone numbers, addresses, and names to standardised formats.
2. **Handling Variability**: Addressing variations in data presentation by applying flexible parsing rules that can adapt to different HTML structures.

D. Feature Selection

Feature selection focuses on identifying and extracting key attributes that will be stored and analysed. This includes:

1. Key Attributes: Extracting essential features such as user names, contact numbers, addresses, and any additional metadata.
2. Attribute Transformation: Transforming attributes into formats suitable for storage and analysis, such

as splitting addresses into components or normalizing phone numbers.

E. Data Storage

Data storage involves saving the transformed data in a reliable and accessible format. This stage includes:

1. **Storing Data in Amazon S3**: Using Boto3 to upload the cleaned and transformed data to an Amazon S3 bucket, ensuring scalability and durability.
2. **Creating a CSV File**: Generating a CSV file of the user details for easy access and analysis

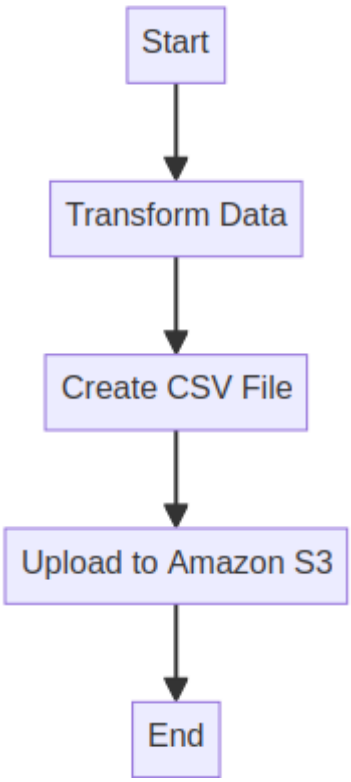
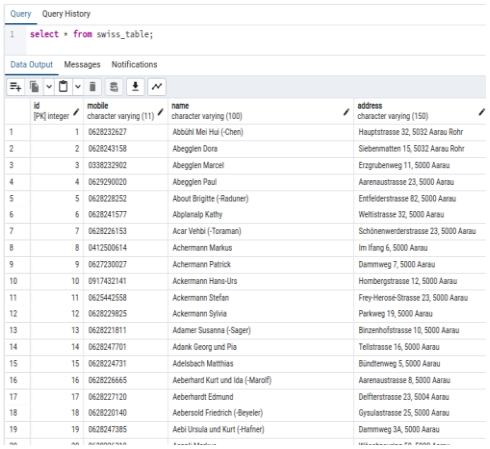


Fig. Data Pipeline Process

Results

The pipeline successfully scrapes user details from multiple city pages on www.local.ch and stores the data in Amazon S3. The use of asynchronous programming significantly reduced the time required for the scraping process, allowing the pipeline to handle a large number of requests efficiently.



```
select * from swiss_table;
```

Data Output | Messages | Notifications

	id [PK] integer	mobile character varying (11)	name character varying (100)	address character varying (150)
1	1	0628232627	Abbühl Mei Hui (-Chen)	Hauptstrasse 32, 5032 Aarau Rohr
2	2	0628243158	Abegglen Dora	Siebenmatten 15, 5032 Aarau Rohr
3	3	0338232902	Abegglen Marcel	Erzgrubenweg 11, 5000 Aarau
4	4	0629290020	Abegglen Paul	Aarenaustrasse 23, 5000 Aarau
5	5	0628228252	About Brigitte (-Raduner)	Entfelderstrasse 82, 5000 Aarau
6	6	0628241577	Abplanalp Kathy	Wellistrasse 32, 5000 Aarau
7	7	0628226153	Acar Vehbi (-Toraman)	Schönerwerderstrasse 23, 5000 Aarau
8	8	0412500614	Achermann Markus	Im Ifang 6, 5000 Aarau
9	9	0627230027	Achermann Patrick	Dammweg 7, 5000 Aarau
10	10	0917432141	Ackermann Hans-Urs	Hombergstrasse 12, 5000 Aarau
11	11	0625442558	Ackermann Stefan	Frey-Herosé-Strasse 23, 5000 Aarau
12	12	0628229825	Ackermann Sylvia	Parkweg 19, 5000 Aarau
13	13	0628221811	Adamer Susanna (-Sager)	Binzenhofstrasse 10, 5000 Aarau
14	14	0628247701	Adank Georg und Pia	Tellstrasse 16, 5000 Aarau
15	15	0628224731	Adelsbach Matthias	Bündtenweg 5, 5000 Aarau
16	16	0628226665	Aeberhard Kurt und Ida (-Marolf)	Aarenaustrasse 8, 5000 Aarau
17	17	0628227120	Aeberhardt Edmund	Delfterstrasse 23, 5004 Aarau
18	18	0628220140	Aebersold Friedrich (-Beyeler)	Gysulastrasse 25, 5000 Aarau
19	19	0628247385	Aebi Ursula und Kurt (-Hafner)	Dammweg 3A, 5000 Aarau

Furthermore, a visual representation of the saved data was included, enhancing the comprehensibility and

accessibility of the stored information. This addition not only provides a clear snapshot of the dataset but also facilitates easier interpretation and analysis of the extracted user details.

Conclusion

The proposed ETL pipeline is designed to efficiently and accurately scrape user details from www.local.ch. By leveraging asynchronous programming, robust HTML parsing, and thorough data preprocessing,the pipeline ensures high-quality data extraction suitable for various applications.The methodology outlined provides a structured approach to handling the complexities of web scraping and data processing, ensuring reliability and scalability.

References

1. Akidau, T., Bradshaw, R., Chambers, C., Chernyak, S., Fernández-Moctezuma, R. J., Lax, R., ... & Whittle, S. (2015). The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. Proceedings of the VLDB Endowment, 8(12), 1792-1803.
2. Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.
3. Marz, N., & Warren, J. (2015). Big Data: Principles and best practices of scalable real-time data systems. Manning Publications.
4. Kreps, J. (2014). Questioning the Lambda Architecture. [Online Article]. Retrieved from https://www.oreilly.com/radar/questioning-the-lambda-architecture/
5. Guhathakurta, A., Boyd, C., & Laing, C. (2017). Data Ingestion Using Apache NiFi. Proceedings of the Practice and Experience on Advanced Research Computing, 1-8.
6. Beauchemin, M. (2015). The rise of Apache Airflow. [Online Article]. Retrieved from https://airflow.apache.org/
7. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., ... & Stoica, I. (2010). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, 15-28.