ETL Notes

Chuks Okoli

02 November, 2023

Extract, Transform and Loading

Data that is collected from multiple sources and combined to become a single source of information, is done through a type of data collection called Batch processing. Extract, Transform and Load or (ETL) is a type of batch processing. **ETL** is the process of extracting large amounts of data from multiple sources and formats and transforming it into one specific format before loading it into a database or target file.

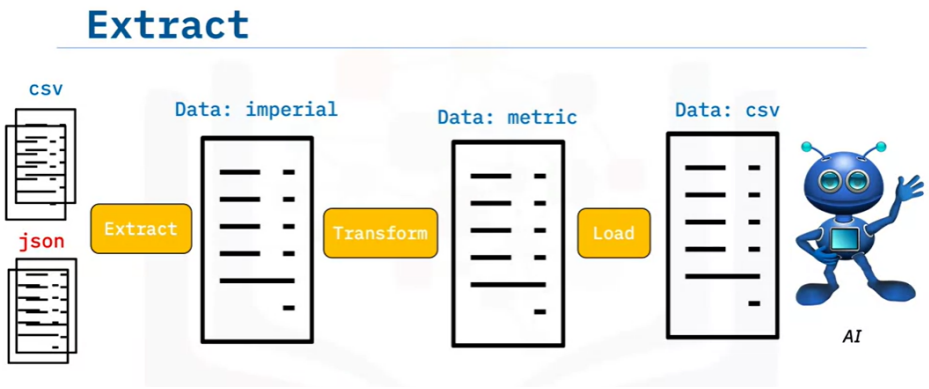

Figure 1: ETL Process

Let's use the following block diagram to represent the ETL pipeline. For example, let's say you are the owner of a start-up that has built an AI to predict if someone is at Risk for diabetes using height and body weight. Some of your data is in CSV format, the other data is JSON files. You need to Extract all this data into one file so the AI can read it. Your data is in imperial but the AI uses metric units so you must transform it. The AI only reads CSV data in one big file, so you need to load The data as CSV Let's implement the following ETL with Python. Let's look at the extraction Step, with some simple examples. Before we discuss the extract function, let's discuss its composite functions. Let's discuss, the glob function, from the glob module. The input is the file extension and the output is a list of files with that particular extension. Consider the list of of .json and .csv files. The input is the glob file extension preceded by a star and a Dot. The output is a list of .csv files. We can do the same for .json files.

```
import glob

# csv files
list_csv = glob('*.csv')
list_csv:['source1.csv', 'source2.csv', 'source3.csv']

# json files
list_json = glob('*.json')
list_json:['source1.json', 'source2.json', 'source3.json']
```

We can write a file to extract CSV of names, height, and weight. The input is the file name of the .csv file, the result is a data frame. We can do the same for JSON formats. The input is the name of the .json file, the output is a data frame. Check out the lab for more examples The Function extract will extract large amounts of data from multiple sources in batches. We create an empty data frame. The Result is an empty data frame with the column names assigned.

```
# Extraction
def extract_from_csv(file_to_process):
    dataframe = pd.read_csv(file_to_process)
    return dataframe

def extract_from_json(file_to_process):
    dataframe = pd.read_json(file_to_process, lines=True)
    return dataframe

def extract_from_xml(file_to_process):
    dataframe = pd.DataFrame(columns=["name", "height", "weight"])
    tree = ET.parse(file_to_process)
    root = tree.getroot()
    for person in root:
        name = person.find("name").text
        height = float(person.find("height").text)
        weight = float(person.find("weight").text)
        dataframe = pd.concat([dataframe, pd.DataFrame([{"name":name, "height":height,
↪   "weight":weight}])], ignore_index=True)
    return dataframe
```

Let's look at the process to extract the .csv file. We start by adding a loop to find all of the CSV files. This can be accomplished easily by adding a glob function. By adding this function it will now find and load all of the CSV file names and with each iteration of the loop, the CSV files are appended to the date frame the first iteration would be appended and then the second iteration resulting in a list of extracted data. We will review the Parameter ignore_index="True" later. This section of code appends the JSON files to the data frame. Let's see how the parameter "ignore index" sets the index in the order each column was appended to the data frame. We will look at the output extracted data. If we did not set the parameter "ignore_index" to true, then the index of the data frame "extracted_data" would be the same as the row Number of the original file. If we set the "ignore index" to true, then the order of each row would be the same as the order the row was appended to the data frame. Now that we have collected the data, the next step in the process is the "Transform" step. This function will convert the column height, which is in inches to millimeters, and the column pounds to kilograms, and return the output in the variable data. Let's take a closer look. In the input data frame, the column height is in feet. We apply the following operation to the column to convert it to meters and round it to two decimal places. The resulting data frame column height is now in meters. We can do the same for weight. Let's look at loading and logging. Now that we have collected and defined the data, it is time to load it into the targetfile. In this case, we save the pandas data frame as a CSV. We have now gone through the process of how to extract, transform, and load data from multiple sources and finalize it into one final targetfile. Before completing our task, we need to create a logging entry. To do this, we will create a logging function. Start by creating a variable named timestamp_format which determines the formatting of the time and date. Next is the "now" variable and it will capture the current time by calling datetime. With this addition, datetime was imported for it to work correctly. We then pull that information together by opening a file and writing the information to the file. With the addition of the "a" all of the data written will be appended to the existing information. By creating this type of entry we are then able to attach a timestamp to each part of the process of when it begins and when it has completed. Now that we have defined all of the code necessary to run the ETL process on the selected data, the last thing we need to

do is call all of functions. We first start by calling the extract_data function. The data received from this step will then be transferred to the second step of transforming the data. After this has completed the data is then loaded into the targetfile. Also, note that before and after every step the time and date for start and completion has been added.

Summary

- ETL is the process of extracting large amounts of data from multiple sources and formats and transforming it into one specific format before loading it into a database or target file.
- Web scraping is a process that can be used to extract information automatically from a website using the two Python modules "Requests" and "BeautifulSoup".
- Use the xml library to parse XML, and the pandas library to parse CSV and JSON data.
- Load a Pandas data frame to an SQL database object using the to_sql()method.
- Use the Pandas read_sql() method to query a database table. This function returns a Pandas data frame with the output to the query.
- Processed data can be stored as a table in a database and retrieved from it using queries, with the Pandas and SQLite libraries.
- Use the xml library to parse XML, and the pandas library to parse CSV and JSON data.
- We learnt:
  - How to write a simple Extract function.
  - How to write a simple Transform function.
  - How to write a simple Load function.
  - How to write a simple Logging function.