

Date: 19th December 2022

Course: Advance Database Systems

Project supervisor:

Dr Joann J. Ordille

# ETL Pipeline for Keyword Extraction

Author: Shashank Gupta

## Contents

<b>1</b>	<b>Abstract</b>	<b>1</b>
<b>2</b>	<b>Introduction</b>	<b>2</b>
<b>3</b>	<b>ETL Architecture</b>	<b>3</b>
<b>4</b>	<b>Methodology</b>	<b>4</b>
4.1	Input . . . . .	5
4.2	Web Scraping . . . . .	5
4.3	Data Cleaning . . . . .	5
4.4	Keyword Extraction . . . . .	6
4.5	Write-back . . . . .	6
<b>5</b>	<b>Results</b>	<b>6</b>
5.1	Execution Time . . . . .	7
<b>6</b>	<b>Conclusion</b>	<b>8</b>

## 1 Abstract

This document proposes an end-to-end approach for creating an application for keyword extraction leveraging the AWS Web-services. Keywords have several applications in Search Engine Optimization (SEO), Name Entity Recognition, Word Embed-

ding Generation, Text Summarization, etc. The proposed keyword extraction process includes three steps - Web-Scrapped Text collection, Data Cleaning & Transformations, and Write-Back of Keywords. The data source for this project would be the text collected from google search via automated web-scrapping. Data transformations will be performed to clean the text and retrieve the highly frequent and relevant keywords using Text-Rank, KeyBERT, and Yake techniques. The write-back phase writes the extracted keywords to AWS DynamoDB. The ETL architecture is implemented in AWS Cloud by leveraging the cloud services like AWS Lambda, S3, DynamoDB, and CloudWatch. Please follow the [GitHub](#) link to understand Keyword Extractor (Written in Python).

## **2 Introduction**

Understanding how Keyword in a search query maps to the relevant web-pages is one of the most important tasks in the field of Search Engine Optimization, which helps the developer team understand the user intent behind their query and bridges the mapping of keywords to the relevant websites. In Text Summarization, keywords are used as input context from large documents for text summarization purposes. This project aims to create an ETL pipeline for extracting relevant keywords from large documents and storing them in a database for understanding Search Engine Optimization and content ranking mechanism. The following Sections will cover:

- ETL Architecture
- Methodology
- Results
- Future Work

### 3 ETL Architecture

The solution Architecture consists of AWS Cloud Services, including AWS Lambda, S3, and DynamoDB, to build an ETL pipeline. The user will update a CSV with a search query in the S3 bucket which will trigger the Lambda function to run the python-based web scrapper and keyword extractor function. The output of extractor - Keywords will be stored in AWS DynamoDB.

Following is a visual representation of the solution architecture:

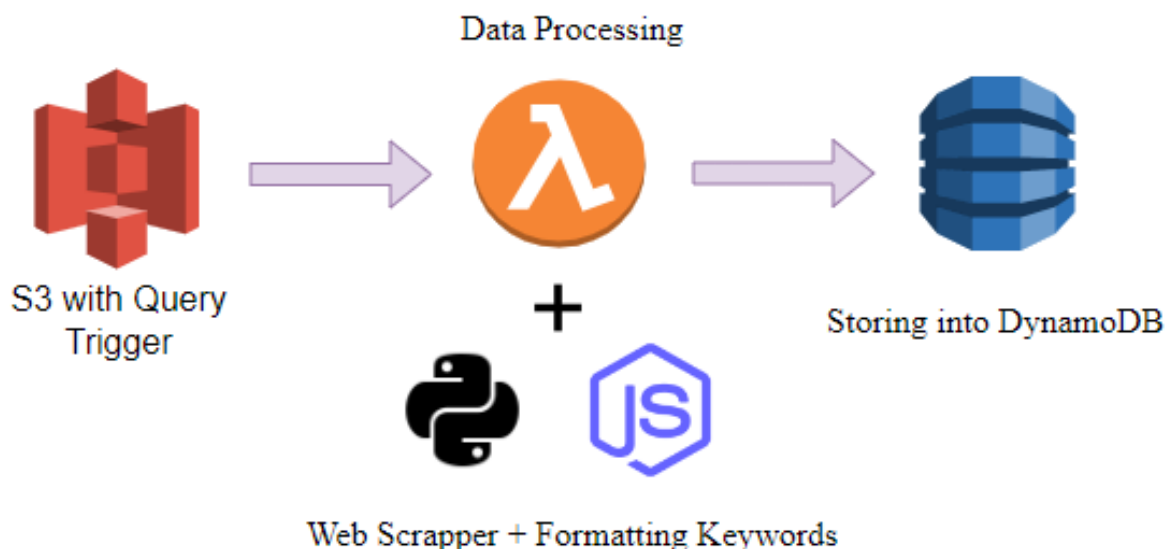


Figure 1: Architecture

This demonstrated architecture leverages the free tier AWS Cloud

Services; hence, the pipeline has further room for improvement in processing power and Read & Write latency. The Python scripts are integrated within the Lambda function for automated cleaning and data loading. It's an essential part of the whole ETL process.

## 4 Methodology

This section explains the steps and approach carried out to build the Keyword extraction utility. Following is the keyword extractor architecture and dataflow diagram.

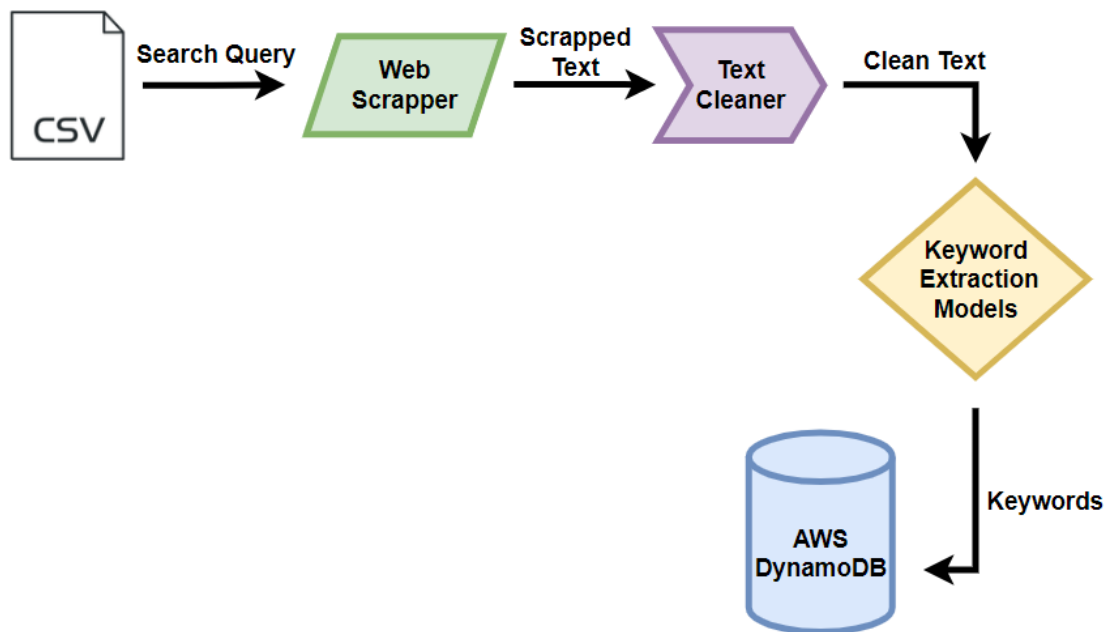


Figure 2: Architecture

The user updates a CSV file with a search query in an S3 bucket similar to the google search query. The updated query in S3 triggers the Lambda function performs Web-scraping, Text-cleaning, Keyword Extraction, and keywords Write-back to AWS DynamoDB

tables. Let's review each section:

#### **4.1 Input**

The following are the parameters required for keyword extraction:

- Queries: Number of queries provided by the user.
- Number of Pages Searched: Number of pages to be searched for each query
- Max Word Limit: max number of words to be extracted
- Grams: Specify grams to be extracted - unigram, bigram, trigram
- Keyword Extraction Model: Specify model needs to be used to extract keywords, options - Yake, Text Rank, KeyBERT

The updated query in S3 triggers the Lambda function, which performs the following operations:

#### **4.2 Web Scraping**

Web Scraper: Web Scraper function takes the user query string as input and leverages the google search from the google python package to get web pages corresponding to it and scrapes all text in the body tag of the website HTML.

#### **4.3 Data Cleaning**

Cleaner: The scrapped web text is passed through a cleaning layer that performs data cleaning tasks like removing symbols, emails, stop-words, and new-line characters. The clean text is

further lemmatized to group the words having inflicted forms together. For example, the words - played playing plays are reduced to the root-word play.

#### **4.4 Keyword Extraction**

Keyword Extractor: Takes clean text as input and returns Keywords (Unigrams, Bigrams, Trigrams). The user needs to specify the grams needed in the output. The script utilizes one out of three pretrained keyword extractor model options available for keyword extraction: KeyBERT, Yake, and Text Rank, out of which Yake returns better results and is more reliable.

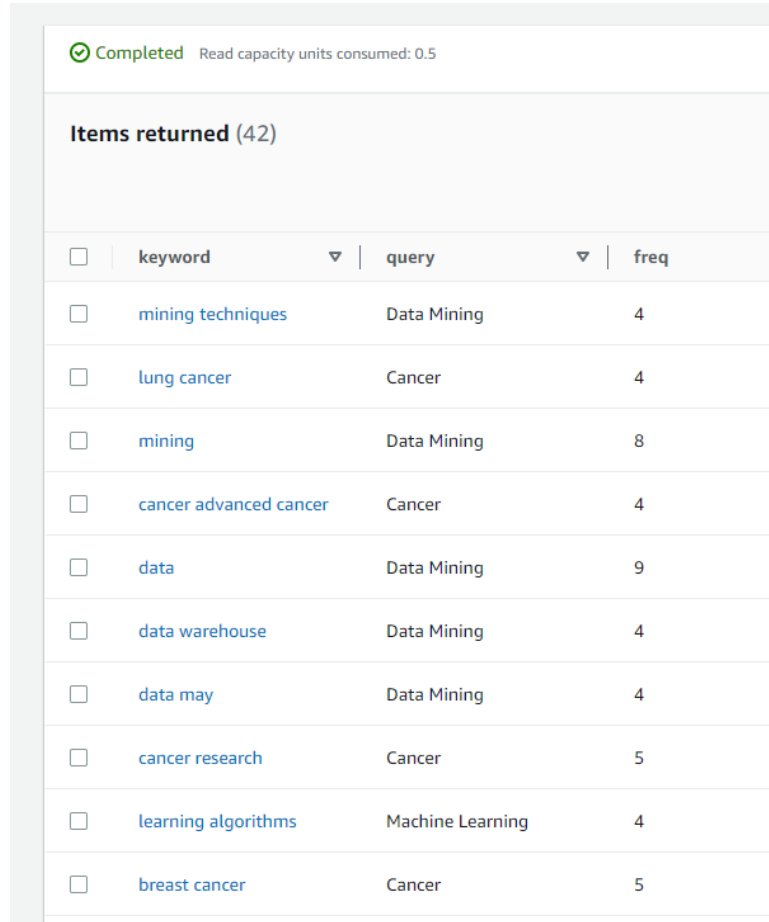
#### **4.5 Write-back**

Once the keywords are extracted, the Python script's driver function will write the output to the DynamoDB tables leveraging the put item method to write it in DynamoDB tables. The keywords will be used as a partition key, and the query string as the sort key.

### **5 Results**

The Result set consists of Unigrams, Bigrams, and Trigrams. A table of scrapped keywords is presented in Figure 2, obtained by running the Web application. For input queries - Machine Learning, Data Mining, and Cancer, we have received 42 Keywords covering unigram, bigram, and trigram. Extraction was performed on average on 3GB RAM and utilized 350 MB of memory. The

run time for this query was 31 seconds.



The screenshot shows a web interface with a status bar at the top indicating 'Completed' and 'Read capacity units consumed: 0.5'. Below this is a section titled 'Items returned (42)'. A table follows, displaying extracted keywords. Each row includes a checkbox, the keyword, a dropdown menu for the query, and the frequency of the keyword.

<input type="checkbox"/>	keyword	query	freq
<input type="checkbox"/>	mining techniques	Data Mining	4
<input type="checkbox"/>	lung cancer	Cancer	4
<input type="checkbox"/>	mining	Data Mining	8
<input type="checkbox"/>	cancer advanced cancer	Cancer	4
<input type="checkbox"/>	data	Data Mining	9
<input type="checkbox"/>	data warehouse	Data Mining	4
<input type="checkbox"/>	data may	Data Mining	4
<input type="checkbox"/>	cancer research	Cancer	5
<input type="checkbox"/>	learning algorithms	Machine Learning	4
<input type="checkbox"/>	breast cancer	Cancer	5

Figure 3: Keywords Extracted

## 5.1 Execution Time

The ETL pipeline is tested against a different number of queries, and the execution time is observed to estimate the efficiency of the pipeline. This analysis will help us understand the expected run time based on the number of queries searched for the pipeline. We can reduce the execution time of the ETL pipeline from a future work perspective using other AWS cloud services like AWS Parallel Cluster, which offers high-performance accelerated computing. Following is a trend observed for execution time when different sets of search queries were given as input:

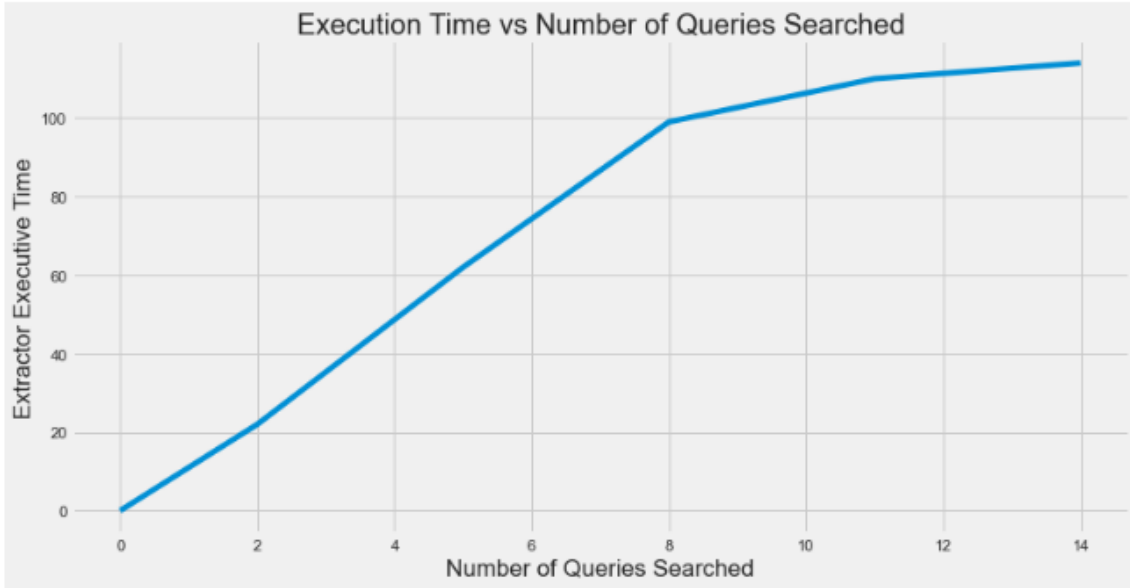


Figure 4: Execution Time vs. Number of Queries Searched

## 6 Conclusion

As expected, the ETL pipeline helped populate the DynamoDB tables with Keywords (Unigrams, Bigrams, and Trigrams). This approach helps create a Keyword database and can be scaled to handle more queries at once. The words extracted using the ETL pipeline were constructive and significant and helped understand the user mindset. The mean execution time of the ETL Pipeline for 3 - 6 queries was 50 seconds. This can be significantly improved using AWS Parallel Cluster.