

German Job Market: Data Engineering project

March 29, 2024



1

Objective : Data extraction/ cleaning

Project repo directory structure

Project Github Repo:

<https://github.com/arunp77/Job-Market-project>

```
Job-Market-project/
|
├── .env # Environment variables file
├── .github/
│   └── workflows/ # GitHub Actions workflow directory
│       └── ci.yml # CI/CD workflow file
├── images/ # Directory for image files
├── scripts/ # Directory for scripts
│   ├── web_scraping/ # Directory for web scraping scripts
│   │   ├── adjurna.py # Script for Adjurna data extraction
│   │   ├── muse.py # Script for Muse data extraction
│   │   └── ss.py # Script for Stepstone data extraction
│   ├── etl/ # Directory for ETL scripts
│   │   └── etlscript.py # ETL script
│   ├── database/ # Directory for database scripts
│   │   └── db_connection.py # Database connection script
│   └── plot_analysis/ # Directory for plot analysis scripts
│       └── uscase.py # Use case plot analysis script
├── data/ # Directory for data
│   ├── scraped_data/ # Directory for scraped data
│   │   ├── adjurna/ # Directory for Adjurna data
│   │   │   └── csv/ # Directory for CSV files
│   │   │       └── adjurna_scraped_data.csv # Adjurna scraped data file
│   │   ├── muse/ # Directory for Muse data
│   │   │   └── csv/ # Directory for CSV files
│   │   │       └── muse_scraped_data.csv # Muse scraped data file
│   │   └── ss/ # Directory for Stepstone data
│   │       └── ss_datascience_germany_20240221.csv # Stepstone data file
│   └── processed_data/ # Directory for processed data
│       ├── adjurna_processed_data/ # Directory for processed Adjurna data
│       │   └── adjurna_scraped_data.csv # Processed Adjurna data file
│       ├── muse_processed_data/ # Directory for processed Muse data
│       │   └── muse_scraped_data.csv # Processed Muse data file
│       └── ss_processed_data/ # Directory for processed Stepstone data
│           └── ss_datascience_germany_20240221.csv # Processed Stepstone data file
├── api.py # FASTapi
├── README.md # Readme file
├── ProjectPlan.md # Project plan file
├── LICENSE.md # License file
├── Contribution-guidelines.md # Contribution guidelines file
└── UserStories.md # User stories file
```

Project repo directory structure

Job-Market-project Public

main 2 Branches 0 Tags

Go to file Add file Code

arunp77 fastapi 210 Commits

File/Folder	Commit Message	Time Ago
.github	Create jekyll-gh-pages.yml	2 days ago
data	fastapi	2 days ago
images	fastapi	yesterday
scripts	fastapi	20 hours ago
.gitignore	Readme.md updated	3 days ago
Contribution-guidelines.md	Update Contribution-guidelines.md	last month
Docker-image-integration.md	Readme.md updated	4 days ago
Dockerfile	fastapi	20 hours ago
FASTapi.md	fastapi	yesterday
LICENSE	Update LICENSE	last month
README.md	fastapi	yesterday
SECURITY.md	Update SECURITY.md	last month
api.py	fastapi	16 hours ago
database-selection.md	Readme.md updated	4 days ago
docker-compose.yml	Readme.md updated	3 days ago
index.html	fastapi	2 days ago
requirements.txt	adding plotly	2 days ago

About

InsightfulRecruit: Unveiling the Job Market Landscape through Data Engineering

[arunp77.github.io/Job-Market-project/](#)

python api docker elasticsearch machine-learning sql mongodb nosql docker-compose databases pyspark apache-kafka apache-airflow dash-plotly fastapi

Readme

GPL-3.0 license

Security policy

Activity

0 stars

2 watching

0 forks

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Contributors 3

arunp77 Arun Kumar Pandey

brindh311

khushboo026

Deployments 9

InsightfulRecruit: Unveiling the Job Market Landscape through Data Engineering

Objective

This project aims to gather, process, create a database and deploy using FASTapi. By the end of the project, we aim to have a clearer understanding of the job market, including sectors with the highest demand, required skills, active cities, and more.

Data Extraction/Processing

Two method of data extraction:

- Connect to API (Adzuna, Muse)
- Web scraping (Stepstone)

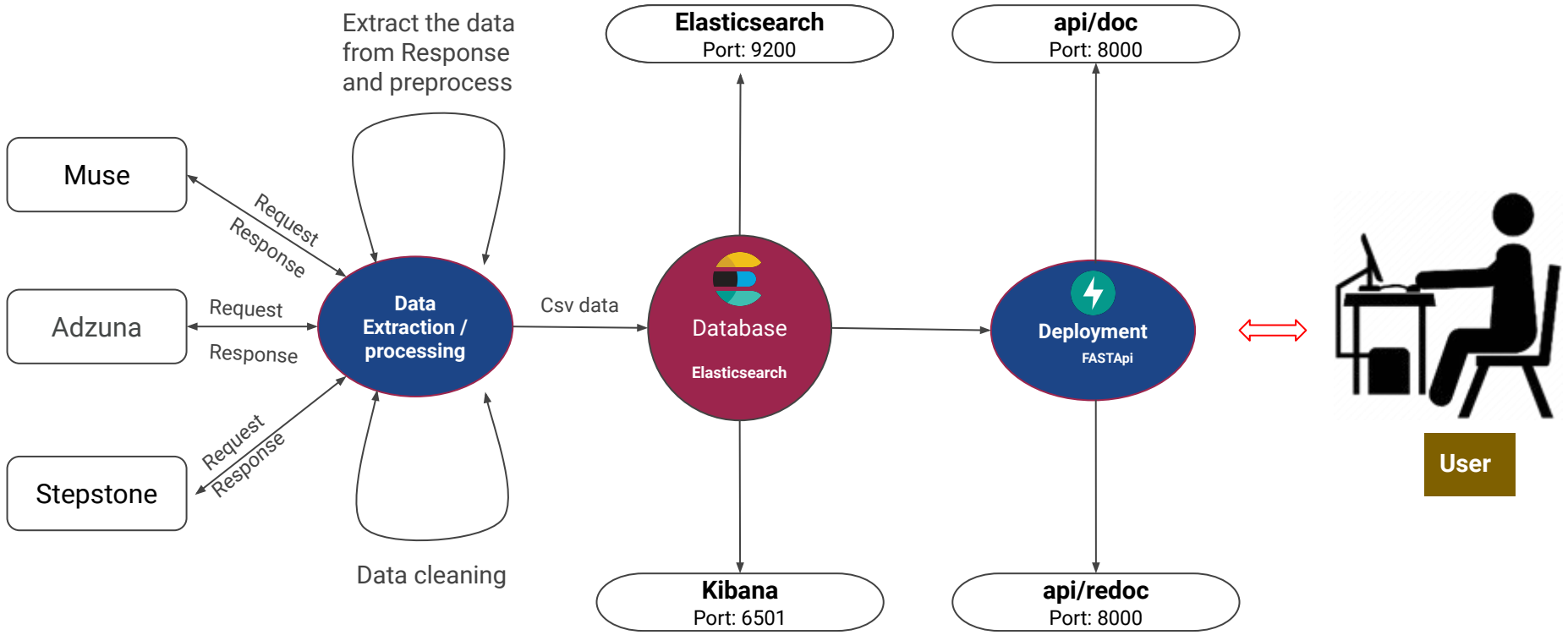
DataBase: Elasticsearch

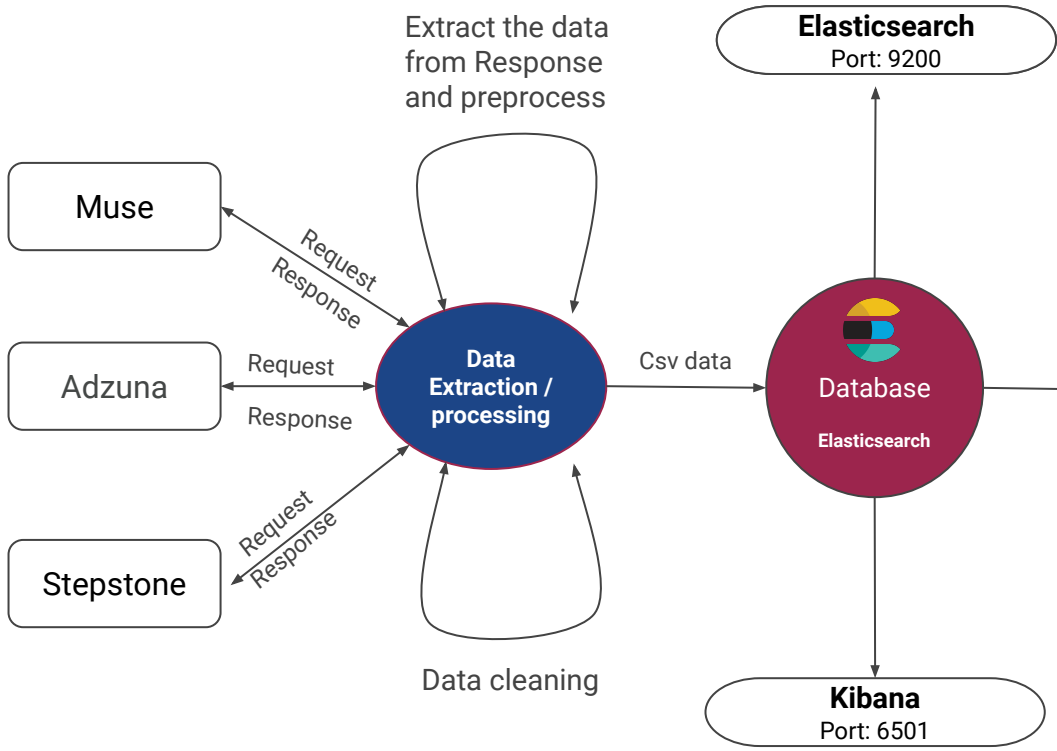
NoSQL DB is better for different schema data from different source

Elasticsearch is better compare to other DB for job market use case, as text search is important.

Deployment: FASTapi

Used FASTapi for deploying our app. The user can call the various APIs using this single app and can see the results.





Database : Query



Job Databases

Fetch



FastAPI
Application

Query

Result
Dataset



Users

Data Extraction : Processing

Data Extraction

Three sources:

- Muse Api
- Adzuna Api
- Web Scraping from Stepstone

Category: Data Science

Data cleaning

- Translation of text from German to English
- Datetime handling (maintaining same date time format in the various data files)

Saving csv format

Saved data format:

- CSV
- JSON

But have used csv for current project. But in future we also create pipeline, which also utilize the json format for further analysis.

Data Extraction : Processing

Data Extraction

Three sources:

- Muse Api
- Adzuna Api
- Web Scraping from Stepstone

Category: Data Science

Data cleaning

- Translation of text from German to English
- Datetime handling (maintaining same date time format in the various data files)

Challenges

APIs: Muse and Adzuna Apis are used to extract the data.

Limitations: Muse has limits on requests/hrs so had to manually set limits on datasets from Muse.

Web Scraping: Stepstone website is dynamic so using beautifulsoup for scraping was not doable. Used selenium to load the page and scrap the data.

For cleaning: the api could not translate the entire text. Have to find out the words and translate them separately in the code.

2

Database : Elasticsearch

Database: Elasticsearch

- **Full-text Search:** Elasticsearch excels in text search tasks, ideal for job market analysis.
- **Scalability and Performance:** Elasticsearch handles large data volumes efficiently, focusing on search performance.
- **Real-time Analytics:** Elasticsearch supports real-time data analysis, crucial for timely insights in job market analysis.
- **Complex Querying and Aggregations:** Elasticsearch offers powerful analytical capabilities for in-depth job market analysis.

Generalized Data Schema

Field Name	Data Type
id	keyword
title	text
company	text
location	text
job_posted	date
categories	text
experience_level	text
full_part_time	text
description	text
link	text
source	text

Example query: Source Stepstone

```
{  
  "_id": "qGgaeo4BdUjihSI212xp",  
  "title": "Data Engineer (m/w/d)",  
  "company": "Hermes Germany GmbH",  
  "location": "Hamburg",  
  "job_posted": "2024-02-17",  
  "Link": "https://www.stepstone.de/stellenangebote--Data-Engineer-m-w-d-Hamburg-Hermes-Germany-GmbH--10751259-inline.html",  
  "source": "stepstone"  
}
```

```
_id: "qGgaeo4BdUjihSI212xp"  
title: "Data Engineer (m/w/d)"  
company: "Hermes Germany GmbH"  
location: "Hamburg"  
job_posted: "2024-02-17"  
link: "https://www.stepstone.de/stellenangebote--Data-Engineer-m-w-d-Hamburg-Hermes-G..."  
source: "stepstone"
```

Example query: Source Adzuna API

```
{  
  "title": "Data Scientist (f/m/d) - database development/BI, engineer",  
  "company": "Capgemini",  
  "location": "München, München (Kreis)",  
  "job_posted": "2024-02-13",  
  "description": "As a data scientist (f/m/d), you are part of our Insights & Data team, which carries out software and consulting projects with a focus on  
data science, analytics, SAP analytics, data warehousing, master data, data quality management and artificial intelligence. We train you in the  
technologies and processes for data analytics and get you ready for your projects. You will be trained and certified by us or our partners. For this position  
in the Defense area there is a...",  
  "link": "https://www.adzuna.de/details/4567064802?utm_medium=api&utm_source=16b1dafa",  
  "source": "adzuna"  
}
```

```
title: "Data Scientist (f/m/d) - database development/BI, engineer"  
company: "Capgemini"  
location: "München, München (Kreis)"  
job_posted: "2024-02-13"  
description: "As a data scientist (f/m/d), you are part of our Insights & Data team, which c...  
link: "https://www.adzuna.de/details/4567064802?utm_medium=api&utm_source=16b1dafa"  
source: "adzuna"
```

Example query: Source Muse API

```
{  
  "title": "Lead Data Science Engineer - Remote US",  
  "company": "Siemens",  
  "location": "Chicago, IL, Flexible / Remote",  
  "job_posted": "2023-05-16",  
  "categories": "Data and Analytics",  
  "experience_level": "Senior Level",  
  "full_part_time": "Full-time",  
  "link": "https://www.themuse.com/jobs/siemens/lead-data-science-engineer-remote-us",  
  "source": "muse"  
}
```

```
title: "Lead Data Science Engineer - Remote US"  
company: "Siemens"  
location: "Chicago, IL, Flexible / Remote"  
job_posted: "2023-05-16"  
categories: "Data and Analytics"  
experience_level: "Senior Level"  
full_part_time: "Full-time"  
link: "https://www.themuse.com/jobs/siemens/lead-data-science-engineer-remote-us"  
source: "muse"
```

Elasticsearch : Launch

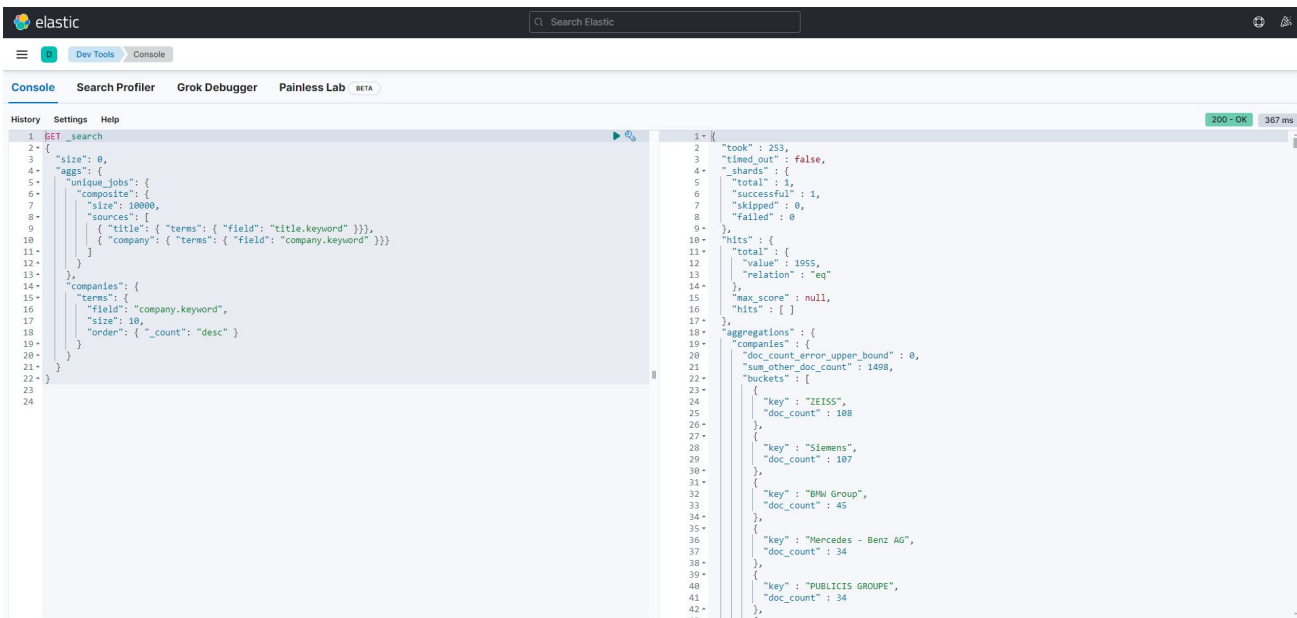
```
$ docker-compose up -d
```

```
PS D:\Arun 2022\Github\Data-engineering-courses\Job-Market-project> docker-compose up -d
[+] Building 170.8s (12/12) FINISHED
=> [python-app internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.15kB
=> [python-app internal] load .dockerignore
=> => transferring context: 2B
=> [python-app internal] load metadata for docker.io/library/python:3.10
=> [python-app auth] library/python:pull token for registry-1.docker.io
=> [python-app internal] load build context
=> => transferring context: 354.63MB
=> [python-app 1/6] FROM docker.io/library/python:3.10@sha256:f9307a98b4ca854bfeb342f7a9c8402557e869a190c4d78ae57157ae82ce8c0d
=> => resolve docker.io/library/python:3.10@sha256:f9307a98b4ca854bfeb342f7a9c8402557e869a190c4d78ae57157ae82ce8c0d
=> => sha256:f9307a98b4ca854bfeb342f7a9c8402557e869a190c4d78ae57157ae82ce8c0d 1.65kB / 1.65kB
=> => sha256:86d840ffceff3d1631aefa53407536bb49b9d22c2ba45f3240f85e6f69ae188a 2.01kB / 2.01kB
=> => sha256:7121d5d5680cf0ab2dccc0e1dd65ed76414e3fb0c294249b5b9319a8fa7c398ae4 49.55MB / 49.55MB
=> => sha256:3cb8f9c23302e175d87a827f0a1c376bd59b1f6949bd3bc24ab8da0d669cdfa0 24.05MB / 24.05MB
=> => sha256:5f899db30843f8330d5a40d1acb26bb00e93a9f21bfff253f31c20562fa264767 64.14MB / 64.14MB
=> => sha256:530eb623b999b117ec9afb986a25b55456028c39df7dd1c7a86a7ebd3d2a95c7 7.34kB / 7.34kB
=> => sha256:567db630df8d441ffe43e050ede26996c87e3b33c99f79d4fba0bf6b7ffa0213 211.14MB / 211.14MB
=> => sha256:d68cd2123173935e339e3feb56980a0aefd7364ad43ca2b9750699e60fbf74c6 6.39MB / 6.39MB
=> => sha256:93260772815949a98e11044f6ca8173565a1b78f170949c5e6def9122a654ccc 17.15MB / 17.15MB
=> => extracting sha256:7121d5d5680cf0ab2dccc0e1dd65ed76414e3fb0c294249b5b9319a8fa7c398ae4
=> => sha256:1f8ac46ddd20addb48905efe7110a45e526466cfa4edaf21a22b0352376c9c55 242B / 242B
=> => sha256:de8f8b8e2a8bccb4bc0ab17ca323c3bedb4035c34b0e73962e2aef7793f1670 3.08MB / 3.08MB
• => => extracting sha256:3cb8f9c23302e175d87a827f0a1c376bd59b1f6949bd3bc24ab8da0d669cdfa0
=> => extracting sha256:5f899db30843f8330d5a40d1acb26bb00e93a9f21bfff253f31c20562fa264767
=> => extracting sha256:567db630df8d441ffe43e050ede26996c87e3b33c99f79d4fba0bf6b7ffa0213
=> => extracting sha256:d68cd2123173935e339e3feb56980a0aefd7364ad43ca2b9750699e60fbf74c6
=> => extracting sha256:93260772815949a98e11044f6ca8173565a1b78f170949c5e6def9122a654ccc
=> => extracting sha256:1f8ac46ddd20addb48905efe7110a45e526466cfa4edaf21a22b0352376c9c55
=> => extracting sha256:de8f8b8e2a8bccb4bc0ab17ca323c3bedb4035c34b0e73962e2aef7793f1670
=> [python-app 2/6] WORKDIR /app
=> [python-app 3/6] COPY . /app
=> [python-app 4/6] COPY requirements.txt .
=> [python-app 5/6] RUN pip install --no-cache-dir -r requirements.txt
=> [python-app 6/6] COPY . .
=> [python-app] exporting to image
=> => exporting layers
=> => writing image sha256:a4b551c6895c6242a3bce2a5d381213cc3d7797cb5e61f6bd8c2a2256428a65c
=> => naming to docker.io/library/job-market-project-python-app
[+] Running 3/3
✓ Container es-container Running
✓ Container job-market-project-python-app-1 Started
✓ Container kb-container Started
```


Elasticsearch : Launch

```
$ docker-compose up -d
```

<http://localhost:5601/>



The screenshot shows the Elasticsearch Dev Tools Console interface. The top bar includes the 'elastic' logo, a search input field, and navigation icons. Below the bar, tabs for 'Console', 'Search Profiler', 'Grok Debugger', and 'Painless Lab' are visible. The 'Console' tab is active, displaying a search query and its results.

Search Query (Left Panel):

```
1 GET _search
2 {
3   "size": 0,
4   "aggs": {
5     "unique_jobs": {
6       "composite": {
7         "size": 10000,
8         "sources": [
9           { "title": { "terms": { "field": "title.keyword" } } },
10          { "company": { "terms": { "field": "company.keyword" } } }
11        ]
12      }
13    },
14    "companies": {
15      "terms": {
16        "field": "company.keyword",
17        "size": 10,
18        "order": { "_count": "desc" }
19      }
20    }
21  }
22 }
```

Search Results (Right Panel):

```
1 {
2   "took": 253,
3   "timed_out": false,
4   "_shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 1955,
13      "relation": "eq"
14    },
15    "max_score": null,
16    "hits": [ ]
17  },
18  "aggregations": {
19    "companies": {
20      "doc_count_error_upper_bound": 0,
21      "sum_other_doc_count": 1498,
22      "buckets": [
23        {
24          "key": "ZEISS",
25          "doc_count": 108
26        },
27        {
28          "key": "Siemens",
29          "doc_count": 107
30        },
31        {
32          "key": "BMW Group",
33          "doc_count": 45
34        },
35        {
36          "key": "Mercedes - Benz AG",
37          "doc_count": 34
38        },
39        {
40          "key": "PUBLICIS GROUPE",
41          "doc_count": 34
42        }
200 - OK 367 ms
```

Elasticsearch : Launch

```
$ docker-compose up -d
```

Challenges

- Choosing the right database is complex due to the multitude of options available in the market.
- For job market searches use cases Elasticsearch is better option,
 - The data received from various sources had different schema
 - we require extensive text analysis capabilities
- Bringing up analytical query have different challenges like data schema text and keyword work differently for same query.

Two use cases

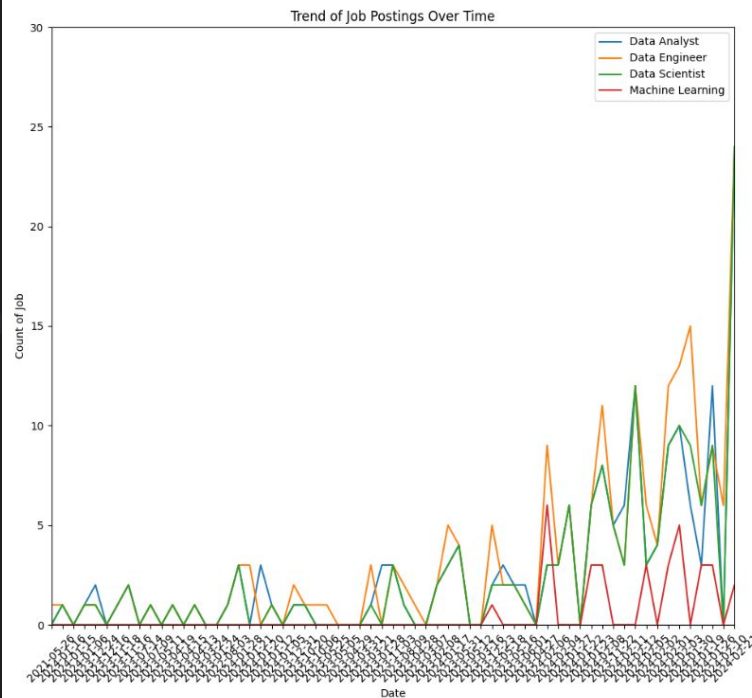
Use case 1

Get the trend of jobs like Data Engineer, Data Scientist, Machine Learning and Data Analyst over time

```
def usecase(usecase):
    if usecase == 1:
        query = {
            "aggs": {
                "job_posted_counts": {
                    "terms": {
                        "field": "job_posted",
                        "size": 200
                    },
                    "aggs": {
                        "data_engineer_count": {
                            "filter": {
                                "match": {
                                    "title": "Data Engineer"
                                }
                            }
                        },
                        "data_scientist_count": {
                            "filter": {
                                "match": {
                                    "title": "Data Scientist"
                                }
                            }
                        },
                        "machine_learning_count": {
                            "filter": {
                                "match": {
                                    "title": "Machine Learning"
                                }
                            }
                        },
                        "data_analyst_count": {
                            "filter": {
                                "match": {
                                    "title": "Data Analyst"
                                }
                            }
                        }
                    }
                }
            }
        }
```

Project use case output:

<https://arunp77.github.io/Job-Market-project/>



Two use cases

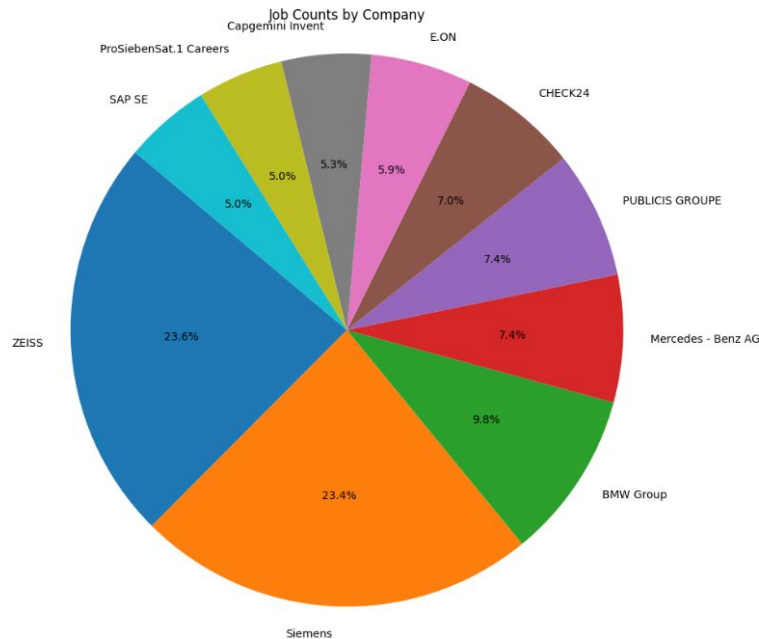
Use case 2

Get the top 10 maximum job posted companies to understand the companies which has max opening in the dataset

```
elif usecase == 2:
    query = {
        "size": 0,
        "aggs": {
            "unique_jobs": {
                "composite": {
                    "size": 10000,
                    "sources": [
                        { "title": { "terms": { "field": "title.keyword" } } },
                        { "company": { "terms": { "field": "company.keyword" } } }
                    ]
                }
            },
            "companies": {
                "terms": {
                    "field": "company.keyword",
                    "size": 10,
                    "order": { "_count": "desc" }
                }
            }
        }
    }
```

Project use case output:

<https://arunp77.github.io/Job-Market-project/>





3

Deployment : FASTApi
and Docker image

FASTApi

Why to choose FASTApi?

- FastAPI offers exceptional performance, outperforming many traditional web frameworks.
- It provides built-in support for asynchronous programming, enabling efficient handling of concurrent requests.
- FastAPI automatically generates interactive API documentation, reducing the need for manual documentation efforts.
- It offers native support for type checking and validation, enhancing code reliability and reducing the likelihood of errors.
- FastAPI seamlessly integrates with existing Python libraries and frameworks, facilitating rapid development and deployment of APIs.
- With its intuitive syntax and declarative approach, FastAPI makes it easy to build scalable and maintainable web applications.

FASTApi

Run the server with:

```
$ uvicorn api:api --reload
```

api.py

Api object in api

<http://localhost:8000/api/docs>

Creates a new instance of the FastAPI class

Create a global Elasticsearch connection

Different Api endpoints

api.py

```
from fastapi import FastAPI
import uvicorn

api = FastAPI(
    title="Job Market API",
    description="API powered by FastAPI for our project.",
    version="1.0.1",
    docs_url="/api/docs",
    redoc_url="/api/redoc"
)

# Create a global Elasticsearch connection
es = db_connection()

@api.get("/")
def welcome():
    return {"message": "Welcome to the Job Market API!"}

@api.get("/load_data")
def load_data():
    return {"error": "Failed to load data into Elasticsearch."}

@api.get("/usecase1")
def get_usecase1():
    return {"error": "Failed to execute use case 1."}

@api.get("/usecase2")
def get_usecase2():
    return {"error": "Failed to execute use case 2."}

if __name__ == "__main__":
    uvicorn.run(api, host="0.0.0.0", port=8000)
```

Interactive OpenAPI docs

Alternative docs

FASTApi

Run the server with:

```
$ uvicorn api:api --reload
```

api.py

Api object in api

<http://localhost:8000/api/docs>

```
@api.get("/", description="Root endpoint to welcome users.")
def welcome() -> dict:
    """Root endpoint to welcome users."""
    return {"message": "Welcome to the Job Market API!"}
```

Create a global
Elasticsearch
connection

Different Api
endpoints

api.py

```
from fastapi import FastAPI
```

```
FastAPI for our project.",
```

Interactive OpenAPI docs

Alternative docs

```
@api.get("/")
def welcome():
    return {"message": "Welcome to the Job Market API!"}

@api.get("/load_data")
def load_data():
    return {"error": "Failed to load data into Elasticsearch."}

@api.get("/usecase1")
def get_usecase1():
    return {"error": "Failed to execute use case 1."}

@api.get("/usecase2")
def get_usecase2():
    return {"error": "Failed to execute use case 2."}

if __name__ == "__main__":
    uvicorn.run(api, host="0.0.0.0", port=8000)
```


FASTApi

Run the server with:

```
$ uvicorn api:api --reload
```

api.py

Api object in api

<http://localhost:8000/api/docs>

Job Market API 1.0.1 OAS 3.1

/openapi.json

API powered by FastAPI for our project. This project aims to gather, process, create a database and deploy using FASTapi. By the end of the project, we aim to have a clearer understanding of the job market, including sectors with the highest demand, required skills, active cities, and more.

default

GET	/	Welcome	⌵
GET	/load_data	Load Data	⌵
GET	/usecase1	Get Usecase1	⌵
GET	/usecase2	Get Usecase2	⌵

FASTApi

Run the server with:

```
$ uvicorn api:api --reload
```

api.py

Api object in api

Challenges

- Integrating existing database connections and use case scripts into the FastAPI framework
- Ensuring familiarity with the FastAPI framework and API **conventions**
- Handling data serialization complexities for efficient data exchange
- Designing and documenting API endpoints effectively
- Implementing robust error handling mechanisms for data integrity

Docker Image

https://hub.docker.com/r/arunp77/job_market

Dockerfile

```
# Layer 1
# Use an official Python runtime as a parent image
FROM python:3.10

# Layer 2
# Copy the current directory contents into the container at /app
COPY requirements.txt /app/

# Layer 4
# Set the working directory in the container
WORKDIR /app

# Layer 5
# Install any needed dependencies specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Layer 6
# Copy the entire project directory into the container
COPY . /app/

# Layer 11
# Run script or command to start the application
ENTRYPOINT ["python", "api.py"]
```

arunp77 / job_market

Contains: Image • Last pushed: 2 minutes ago

Scout active ☆ 0

90

Public

Docker Image

https://hub.docker.com/r/arunp77/job_market

Docker credentials

ci-cd.yml

```
- name: Build Docker image
  run: |
    docker build -t arunp77/job_market:latest .

- name: Push Docker image
  run: |
    echo "${ secrets.DOCKER_PASSWORD }}" | docker login -u "${ secrets.DOCKER_USERNAME }}" --password-stdin
    docker push arunp77/job_market:latest
```

arunp77 / job_market

Contains: Image • Last pushed: 2 minutes ago

Scout active ☆ 0

90

Public



4

Project Showcase: Live Demonstration

Project use case output:

<https://arunp77.github.io/Job-Market-project/>

Demonstration video



JOB MARKET: DATA ENGINEERING



ETL: Extract Transform & Load

Python, Elasticsearch, Docker, FastApi



Limitations & Future Directions

Limitations



The quality of the data can be improved by web scraping from multiple sources of websites. But most of the sites don't allow scraping. We need to get permission.



Many APIs have rate limit which limits the data we can access from that site.



We are considering only few sources with few filters for job posting. We may not get all the data related to that posting.

Future Direction



Machine learning: We plan to work on 'Job title classification', 'Location based analysis', 'Job prediction'

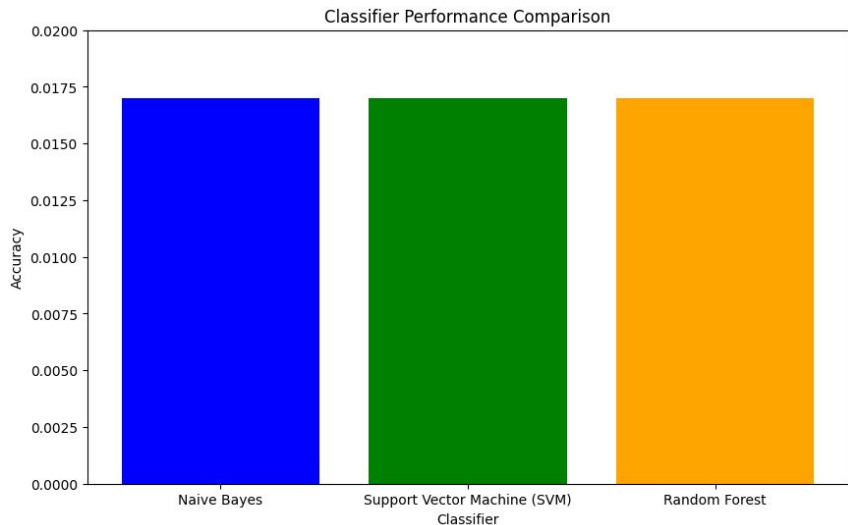


Full Automation : Airflow



Add unit test and integrate with git CI/CD and then Deploy the final docker container in cloud and make available for global use.

Future direction in progress: Machine learning



JOB Title classification:

- **Task: 1** Classify job titles into predefined categories or clusters based on their descriptions or titles.
- **Algorithms:** Text Classification (Naive Bayes, Support Vector Machines (SVM), Random Forests)

Job Posting Frequency Prediction:

- **Task: 2** Identify the most important features that influence job posting characteristics like job title, location, etc.
- **Algorithms:** Random Forests: Feature importances analysis.

Thank

