DSI321 Project

Overview

This project is designed to monitor and analyze public discussions about Thammasat University (TU) using real-time data scraping and natural language processing (NLP). The system begins with web scraping to extract posts and comments related to TU, then processes this content to generate a word cloud that highlights the most prominent keywords. The entire process is orchestrated and automated using Prefect to ensure consistent data updates and analysis.

Key capabilities include:

- Real-time scraping of social media posts and news articles mentioning TU
- NLP processing to extract and display significant terms
- Visualization using word clouds
- CI/CD integrations to ensure data quality and maintain code security

Tools Used

Tool	Purpose
lakeFS	Acts as a data versioning system, ensuring reproducibility and control over all changes in the dataset
Docker	Containerizes the application and its dependencies, enabling seamless deployment across different environments
Prefect	Orchestration tool to automate and schedule the scraping and processing pipelines
Streamlit	Used to create an interactive web-based dashboard for visualizing word clouds and key metrics

Project Structure

```
.
├── config
│   ├── docker
│   │   ├── Dockerfile.cli
│   │   └── Dockerfile.worker
│   ├── logging
│   │   └── modern_log.py
│   └── path_config.py
├── src
│   ├── backend
│   │   ├── load
│   │   │   └── lakefs_loader.py
│   │   ├── pipeline
│   │   │   ├── incremental_scrape_flow.py
│   │   │   └── initial_scrape_flow.py
│   │   ├── scraping
│   │   │   ├── x_login.py
│   │   │   └── x_scraping.py
│   │   └── validation
│   │       └── validate.py
│   └── fronend (typo, should be "frontend")
│       └── streamlit.py
├── test
├── .env.example
├── .gitignore
├── README.md
├── docker-compose.yml
├── pyproject.toml
├── requirements.txt
└── start.sh
```

Schema

This project enforces a strict schema and data validation protocol to ensure data consistency and integrity. Below is the schema and the validation results from the processed dataset (`data.parquet`):

DataFrame Schema (`df_verlify.dtypes`):

Column	Data Type
category	string[python]
tag	string[python]
username	string[python]
tweetText	string[python]

timestamp	datetime64[ns, UTC]
scrapeTime	datetime64[ns]
tweet_link	string[python]
index	int64
year	int32
month	int32
day	int32

Schema Validation Summary

Check	Result
Schema matches original format	✅ `True`
Number of records > 1000	✅ `True`
Duplicate records	✅ `0`
Null values in all columns	✅ `0`
Data types are consistent	✅ `True`

Benefits

Educational Benefits

- Hands-on experience in real-time data pipeline development
- Practice with Docker, Prefect, and Streamlit in production settings
- Application of CI/CD and data validation using GitHub Actions

Practical Benefits

- Reusable template for social media monitoring and keyword analysis
- Supports real-time, incremental scraping flows
- Easy to scale and deploy in both local and cloud environments

Organizational Benefits

- Validated data ensures insights are reliable and reproducible
- Automation reduces the need for manual monitoring
- Can be adapted to other sentiment or public opinion use cases

Prepare

1. Create a virtual environment

```
python -m venv .venv
```

2. Activate the virtual environment

- Windows

```
source .venv/Scripts/activate
```

- macOS & Linux

```
source .venv/bin/activate
```

3. Run the startup script

```
bash start.sh
# or
./start.sh
```

Running Prefect

1. Start the Prefect server

```
docker compose --profile server up -d
```

2. Connect to the CLI container

```
docker compose run cli
```

3. Run the initial scraping flow (to collect all tweets for base data)

```
python src/backend/pipeline/initial_scrape_flow.py
```

4. Schedule scraping every 15 minutes (incremental updates)

```
python src/backend/pipeline/incremental_scrape_flow.py
```

- **View the Prefect flow UI** Open your browser and go to: http://localhost:42000