Enhancements in Algorithms and Data Structures

Zaffar Shiekh

CS-499-10941-M01 Computer Science Capstone 2024 C-4

Federico Bermudez

08/18/2024

**1. Introduction**

This document focuses on the enhancements made to the Grazioso Salvare Dashboard concerning algorithms and data structures. These updates were implemented to improve the efficiency of data retrieval and processing within the dashboard, ensuring that it can handle larger datasets and deliver faster responses.

**2. Software Requirements**

Before implementing the changes, ensure that the following software is installed:

1. **Python 3.x**
2. **Jupyter Lab** or **Visual Studio Code (VSCode)** with Python extensions
   * **Jupyter Lab**: Install using pip:

pip install jupyterlab

* + **VSCode**: Install from [Visual Studio Code's official website](https://code.visualstudio.com/) and add the Python extension.

1. **MongoDB Community Edition**
   * Install from [MongoDB's official website](https://www.mongodb.com/try/download/community).
2. **MongoDB Compass** (optional but recommended)
   * Install from [MongoDB Compass](https://www.mongodb.com/try/download/compass).
3. **Required Python Packages**
   * Install via pip:

pip install pandas dash dash\_table pymongo

**3. File Setup**

Ensure the following files are in place:

1. **Jupyter Notebook (example\_notebook.ipynb)** or Python script (example\_script.py)\*\*
   * Store this file in the main project directory.
2. **Data File (AAC.animals.json)**
   * Place this file in the same directory as the Jupyter Notebook or Python script.

**Directory Structure Example**:

GraziosoSalvareProject/

│

├── example\_notebook.ipynb # or example\_script.py

└── AAC.animals.json

**4. Implementing Algorithm and Data Structure Enhancements**

**4.1 Old Code**

**Original Filtering Logic:**

@lru\_cache(maxsize=32)

def get\_filtered\_data(filter\_type):

if filter\_type == 'WR':

return pd.DataFrame(list(collection.find({"rescue\_type": "Water Rescue"})))

elif filter\_type == 'MWR':

return pd.DataFrame(list(collection.find({"rescue\_type": "Mountain Rescue"})))

elif filter\_type == 'DIT':

return pd.DataFrame(list(collection.find({"rescue\_type": "Disaster Tracking"})))

return df

**Issues:**

* The filtering logic was very basic and did not account for multiple relevant fields (e.g., animal\_type, breed) or more complex search patterns.
* The "Disaster or Individual Tracking" filter did not correctly capture all relevant cases, leading to incomplete results.

**4.2 New Code**

**Updated Filtering Logic:**

from functools import lru\_cache

from dash.dependencies import Input, Output

import pandas as pd

@lru\_cache(maxsize=32)

def get\_filtered\_data(filter\_type):

if filter\_type == 'WR':

return pd.DataFrame(list(collection.find({"breed": {"$regex": "Labrador Retriever|Newfoundland", "$options": "i"}})))

elif filter\_type == 'MWR':

return pd.DataFrame(list(collection.find({"breed": {"$regex": "German Shepherd|Border Collie", "$options": "i"}})))

elif filter\_type == 'DIT':

return pd.DataFrame(list(collection.find({

"$and": [

{"animal\_type": {"$in": ["Dog", "Cat"]}}, # Include both dogs and cats

{"$or": [

{"outcome\_subtype": {"$regex": "Disaster|Tracking", "$options": "i"}},

{"outcome\_type": {"$regex": "Disaster|Tracking", "$options": "i"}},

{"breed": {"$regex": "Bloodhound|Beagle|Siamese|Persian", "$options": "i"}} # Example breeds for tracking, including cats

]}

]

})))

return df

**Changes and Reasons:**

* **Inclusion of Multiple Fields:** To ensure a more comprehensive search, the filtering logic now checks multiple fields (breed, outcome\_subtype, outcome\_type, animal\_type).
* **Regex and OR Conditions:** The "$regex" and "$or" conditions were added to allow for flexible matching of terms related to "Disaster" and "Tracking," ensuring that the relevant data (both dogs and cats) is captured.
* **Inclusion of Cats:** The animal\_type field is explicitly checked to include both "Dog" and "Cat" in the results, addressing the previous issue where only dogs were returned.

**5. Running the Dashboard**

1. **Open Jupyter Lab or VSCode**:
   * Run Jupyter Lab by executing:

jupyter lab

* + Or open the Python script in VSCode.

1. **Load the Notebook or Script**:
   * Open example\_notebook.ipynb in Jupyter Lab or example\_script.py in VSCode.
2. **Run the Notebook or Script**:
   * Execute all cells in the notebook or run the Python script to launch the dashboard. In VSCode, you can run the script by pressing F5 or using the "Run Python File" command.
3. **Access the Dashboard**:
   * Once the dashboard is running, open your web browser and navigate to http://127.0.0.1:8050/ to view the dashboard interface.

**6. Testing the Enhancements**

* Use the radio buttons to filter data in the dashboard and observe the improved response times due to the algorithm optimizations.
* Ensure that both dogs and cats are included in the "Disaster or Individual Tracking" results.