Proof of Concept: DisasterEye

Real-Time Disaster Information Aggregation Software

---

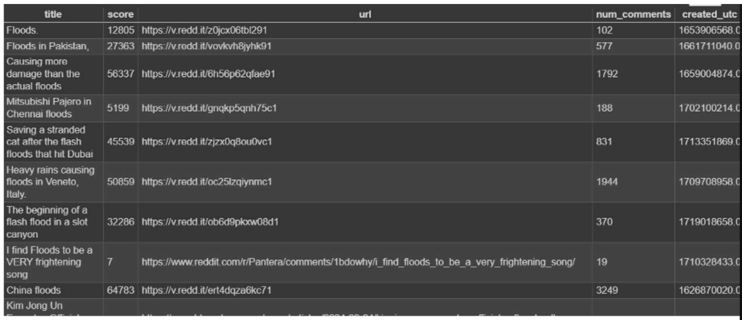
1. Objective

DisasterEye is designed to enhance disaster response efforts by aggregating real-time data from social media, news portals, and other open sources. The platform filters and categorizes the data automatically, making it easier for disaster response teams to prioritize information and respond more effectively. It also includes a critical alert system to notify local authorities in case of high-severity incidents.

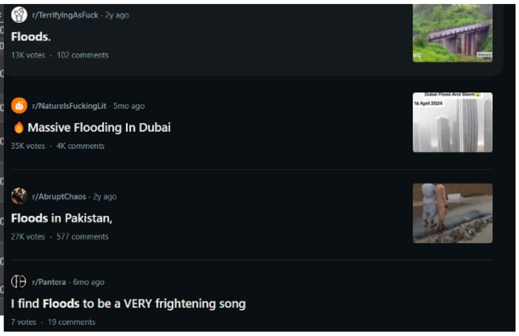
---

2. How It Works

1. **Data Collection (Web Scraper)**
   - A web scraper continuously monitors public social media platforms such as Reddit, Facebook, and Instagram, as well as news websites.
   - It extracts relevant posts based on predefined keywords related to disaster events (e.g., "flood," "earthquake").



title	score	url	num_comments	created_utc
Floods.	12805	https://v.redd.it/z0jcx06tbl291	102	1653906568.0
Floods in Pakistan,	27363	https://v.redd.it/vovkvh8jyhk91	577	1661711040.0
Causing more damage than the actual floods	56337	https://v.redd.it/6h56p62qfae91	1792	1659004874.0
Mitsubishi Pajero in Chennai floods	5199	https://v.redd.it/gnqkp5qnh75c1	188	1702100214.0
Saving a stranded cat after the flash floods that hit Dubai	45539	https://v.redd.it/zjzx0q8ou0vc1	831	1713351869.0
Heavy rains causing floods in Veneto, Italy.	50859	https://v.redd.it/oc25lzqiynmc1	1944	1709708958.0
The beginning of a flash flood in a slot canyon	32286	https://v.redd.it/ob6d9pkxw08d1	370	1719018658.0
I find Floods to be a VERY frightening song	7	https://www.reddit.com/r/Pantera/comments/1bdowhy/i_find_floods_to_be_a_very_frightening_song/	19	1710328433.0
China floods	64783	https://v.redd.it/ert4dqza6kc71	3249	1626870020.0
Kim Jong Un				



r/TerrifyingAsFuck · 2y ago
Floods.
13K votes · 102 comments

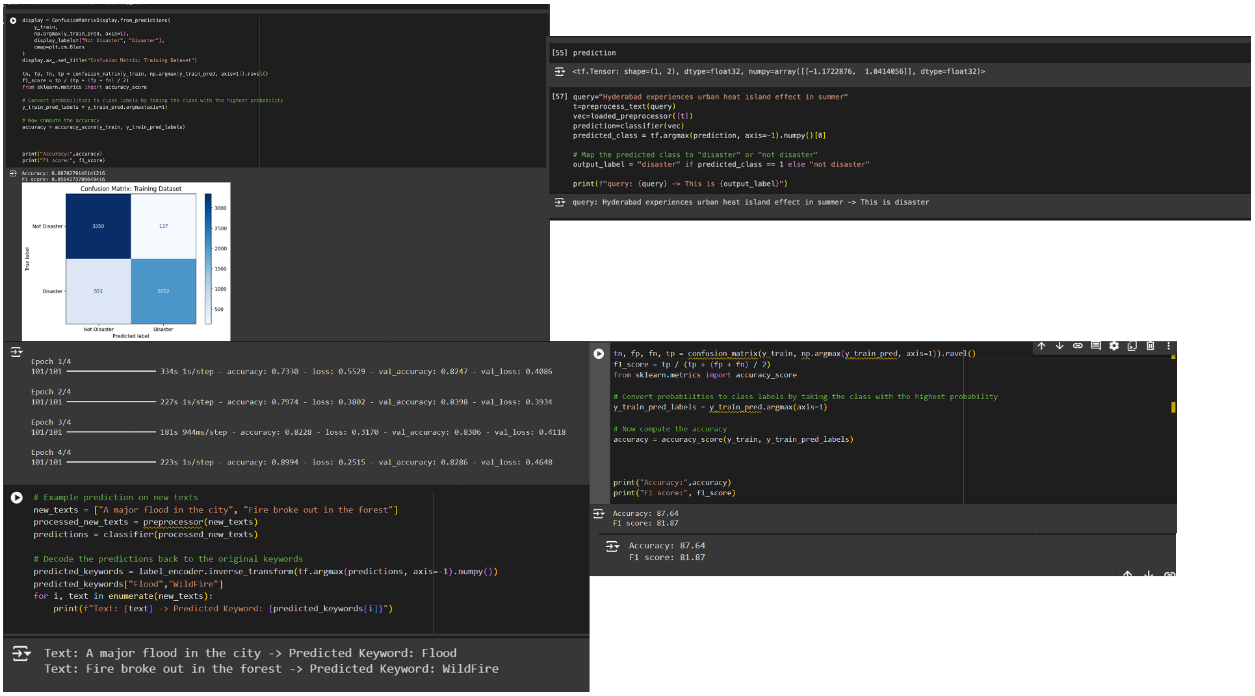
r/NatureIsFuckingLit · 5mo ago
Massive Flooding In Dubai
35K votes · 4K comments

r/AbruptChaos · 2y ago
Floods in Pakistan,
27K votes · 577 comments

r/Pantera · 6mo ago
I find Floods to be a VERY frightening song
7 votes · 19 comments

   -

2. **Data Processing (AI Model)**
   - The scraped data is passed through an AI model powered by Natural Language Processing (NLP).
   - The AI model classifies each post based on **disaster type**, **severity** (low, medium, high), and **location**.
   - This classification ensures that only relevant, actionable information reaches the dashboard, reducing noise and misinformation.



```
display = ConfusionMatrixDisplay.from_predictions(
    y_train,
    np.argmax(y_train_pred, axis=1),
    display_labels=["Not Disaster", "Disaster"],
    cmap=plt.cm.Blues
)
display.ax_.set_title("Confusion Matrix: Training Dataset")

tn, fp, fn, tp = confusion_matrix(y_train, np.argmax(y_train_pred, axis=1)).ravel()
f1_score = tp / (tp + (fp + fn) / 2)
from sklearn.metrics import accuracy_score

# Convert probabilities to class labels by taking the class with the highest probability
y_train_pred_labels = y_train_pred.argmax(axis=1)

# Now compute the accuracy
accuracy = accuracy_score(y_train, y_train_pred_labels)



print("Accuracy:",accuracy)
print("F1 score:", f1_score)
```

```
Accuracy: 0.8870279146141216
F1 score: 0.8564273789649416
```

```
[55] prediction
```

```
<tf.Tensor: shape=(1, 2), dtype=float32, numpy=array([[-1.1722876,  1.0414056]], dtype=float32)>
```

```
[57] query="Hyderabad experiences urban heat island effect in summer"
     t=preprocess_text(query)
     vec=loaded_preprocessor([t])
     prediction=classifier(vec)
     predicted_class = tf.argmax(prediction, axis=-1).numpy()[0]

     # Map the predicted class to "disaster" or "not disaster"
     output_label = "disaster" if predicted_class == 1 else "not disaster"

     print(f"query: {query} -> This is {output_label}")
```

```
query: Hyderabad experiences urban heat island effect in summer -> This is disaster
```

```
Epoch 1/4
101/101 ━━━━━━━━━━━━━━━━━━━━ 334s 1s/step - accuracy: 0.7330 - loss: 0.5529 - val_accuracy: 0.8247 - val_loss: 0.4086

Epoch 2/4
101/101 ━━━━━━━━━━━━━━━━━━━━ 227s 1s/step - accuracy: 0.7974 - loss: 0.3802 - val_accuracy: 0.8398 - val_loss: 0.3934

Epoch 3/4
101/101 ━━━━━━━━━━━━━━━━━━━━ 181s 944ms/step - accuracy: 0.8228 - loss: 0.3170 - val_accuracy: 0.8306 - val_loss: 0.4118

Epoch 4/4
101/101 ━━━━━━━━━━━━━━━━━━━━ 223s 1s/step - accuracy: 0.8994 - loss: 0.2515 - val_accuracy: 0.8286 - val_loss: 0.4648
```

```
tn, fp, fn, tp = confusion_matrix(y_train, np.argmax(y_train_pred, axis=1)).ravel()
f1_score = tp / (tp + (fp + fn) / 2)
from sklearn.metrics import accuracy_score

# Convert probabilities to class labels by taking the class with the highest probability
y_train_pred_labels = y_train_pred.argmax(axis=1)

# Now compute the accuracy
accuracy = accuracy_score(y_train, y_train_pred_labels)



print("Accuracy:",accuracy)
print("F1 score:", f1_score)
```

```
Accuracy: 87.64
F1 score: 81.87
```

```
Accuracy: 87.64
F1 score: 81.87
```

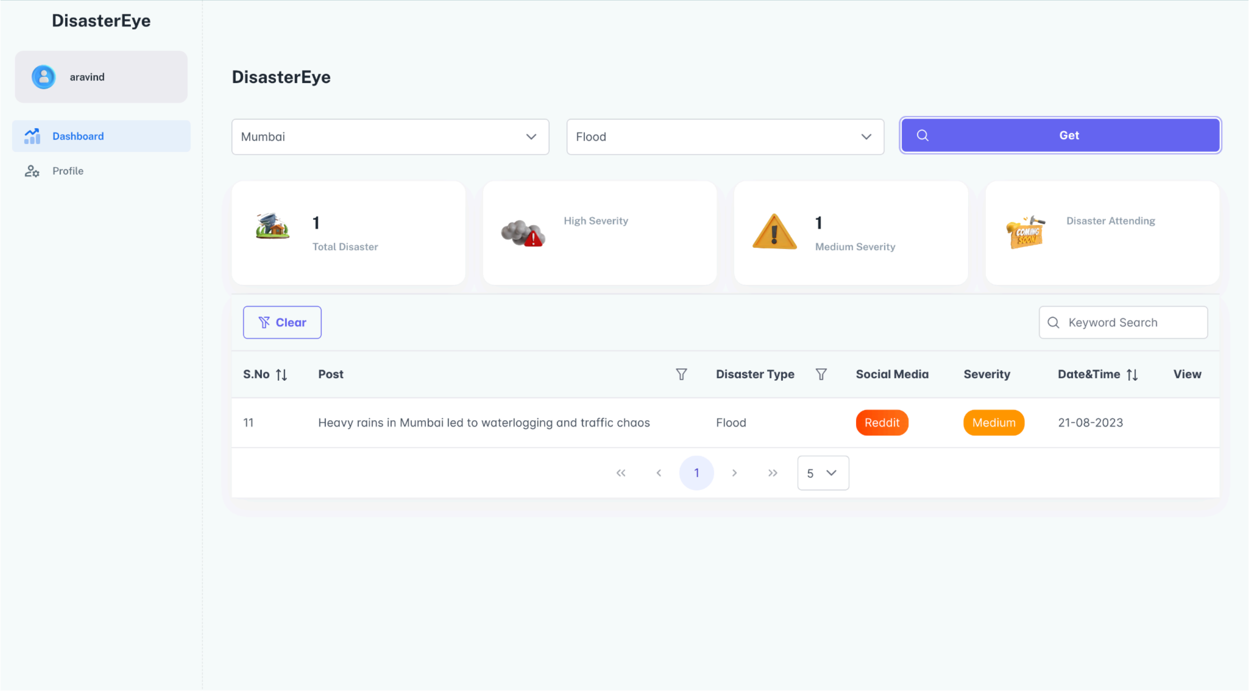
```
# Example prediction on new texts
new_texts = ["A major flood in the city", "Fire broke out in the forest"]
processed_new_texts = preprocessor(new_texts)
predictions = classifier(processed_new_texts)

# Decode the predictions back to the original keywords
predicted_keywords = label_encoder.inverse_transform(tf.argmax(predictions, axis=-1).numpy())
predicted_keywords["Flood","WildFire"]
for i, text in enumerate(new_texts):
    print(f"Text: {text} -> Predicted Keyword: {predicted_keywords[i]}")
```

```
Text: A major flood in the city -> Predicted Keyword: Flood
Text: Fire broke out in the forest -> Predicted Keyword: WildFire
```

3. **Categorization and Filtering**
    - After classification, the data is categorized by:
        - **Disaster Type** (e.g., flood, earthquake)
        - **Severity** (high, medium, low)
        - **Location** (e.g., New York)
    - The categorized data is stored in a database for easy retrieval.
4. **Automatic Alerts for High Severity Disasters**
    - If the AI model detects a **High Severity Disaster**, an automatic alert system is triggered.
    - **Local authorities** will receive real-time notifications via:
        - **SMS**
        - **WhatsApp**
        - **Phone call**
    - These alerts are sent using the **MSG91** API to ensure timely response and coordination.
5. **Dashboard Display**
    - The processed and categorized data is displayed on a user-friendly dashboard.
    - Users can apply filters to search by **city**, **disaster type**, and **severity** to prioritize information that requires immediate attention.
    - The dashboard also highlights key metrics like **Total Disasters**, **High Severity Incidents**, and **Medium Severity Incidents**, helping teams make quick decisions.

Dashboard Screenshot



DisasterEye

Mumbai

Flood

1 Total Disaster

High Severity

1 Medium Severity

Disaster Attending

S.No	Post	Disaster Type	Social Media	Severity	Date&Time	View
11	Heavy rains in Mumbai led to waterlogging and traffic chaos	Flood	Reddit	Medium	21-08-2023	

1 5

3. Key Features

- **Real-Time Data Aggregation**: Gathers disaster-related data from social media platforms automatically.
- **AI-Driven Categorization**: Uses NLP and machine learning to classify disaster posts by type, severity, and location.
- **Custom Filters**: Allows users to search and filter based on disaster type and city to quickly access critical information.
- **Alert System**: For **High Severity Disasters**, sends **SMS**, **WhatsApp**, and **Phone Calls** to local authorities via **MSG91**.

4. Technology Stack

- **Frontend**: React.js
- **Backend**: Node.js/Flask
- **AI Model**: NLP and Machine Learning for real-time classification
- **Database**: MongoDB/PostgreSQL
- **Cloud**: AWS/Azure for scalability
- **Alert System**: MSG91 API for sending critical alerts via SMS, WhatsApp, and Phone Calls

5. Conclusion

DisasterEye has successfully demonstrated its ability to collect, categorize, and present disaster-related data in real time. By automating these processes, the platform significantly improves disaster response times. The integration with MSG91 ensures that local authorities are notified immediately of any high-severity incidents, allowing for a faster and more coordinated disaster response, ultimately saving lives.