Disaster Detection and Classification Using Text, Images, and Real-Time Twitter Scraping

1. Abstract

This project aims to develop an efficient disaster detection system using image and text classification models and a real-time Twitter scraper. The system integrates deep learning techniques for image classification, NLP methods for text classification, and a web-based interface for live disaster monitoring. The project combines disaster tweet and not-spam datasets for text analysis, and disaster image datasets, consolidating fine classes into broad classes to overcome class imbalance issues.

2. Introduction

Problem Statement

Detecting disasters quickly is crucial for effective response, but current systems often fall short. Many solutions rely on either text or image data, failing to combine both for more accurate detection. Additionally, imbalanced datasets and the lack of real-time monitoring from platforms like Twitter lead to delays in disaster response.

Objectives:
The key objectives of this project are:

- Develop a deep learning-based image classification model for disaster detection.
- Implement NLP techniques for classifying disaster-related tweets.
- Create a real-time Twitter scraper to gather and classify live tweets and images.
- Build a web platform for users to upload content and monitor disasters in real-time.

Scope:
The project covers three areas:

- **Text Classification**: Using models like Logistic Regression, Random Forest, and RNN for disaster-related tweet detection.
- **Image Classification**: Using transfer learning with VGGNet for disaster image classification.
- **Real-Time Scraping**: Using Selenium to collect and classify live Twitter data.

Importance

Real-time disaster detection enables faster response, reduces damage, and improves decision-making during crises. Integrating text, images, and live Twitter scraping offers a comprehensive solution for timely disaster management.

3. Dataset Description

Text Data

- **Source**
  - **Disaster Tweet Dataset:** Includes tweets labeled as disaster or not.
  - **Not-Spam Dataset:** Contains non-spam and spam messages for balanced text classification.
- **Preprocessing**
  - **Data Cleaning:** Removed URLs, special characters, and numbers. Converted text to lowercase, removed stopwords (except pronouns like "he," "she," "they"), and applied lemmatization.
  - **Handling Imbalanced Classes:** The class_weight parameter in models like Logistic Regression and SVM was used to assign higher weights to minority classes.
- **Feature Extraction**
  - **TF-IDF (Term Frequency-Inverse Document Frequency):** Applied to convert text into numerical vectors.
  - **Tokenization and Padding:** For RNN/LSTM models, texts were tokenized and padded to a fixed sequence length.

Image Data

- **Source**: The dataset consists of disaster images with both fine-grained and broad classes, enabling the model to distinguish between different disaster types while benefiting from broader categories for better classification performance.
- **Preprocessing**: To address class imbalance, fine classes are consolidated into broader categories (e.g., merging various flood types into a single "Flood" class). This reduces skew in the data distribution, improving training efficiency.
- **Augmentation**: Various data augmentation techniques are applied to enhance training diversity and reduce overfitting, including:
  - **Rotation**: Randomly rotating images up to 40 degrees.
  - **Shifts**: Horizontal and vertical shifts by a fraction of the image dimensions.
  - **Shearing**: Applying shearing transformations.
  - **Zooming**: Randomly zooming into images.
  - **Flips**: Randomly flipping images horizontally.

Twitter Scraping:

The Twitter scraper uses Selenium to automate the browser and extract text and images from tweets based on user-defined hashtags. It initializes a Chrome WebDriver to perform searches and scrolls through the Twitter feed, collecting live data until it reaches the specified number of posts.

Collecting Text and Images Based on Hashtags

1. **Setup**: Users provide a hashtag and the maximum number of tweets to scrape.
2. **Tweet Extraction**: The scraper extracts text and image URLs from tweets containing the specified hashtag.
3. **Image Downloading**: Images are downloaded to local directories, organized by tweet sequence (e.g., Post_1, Post_2).
4. **Text Storage**: The text content is saved in corresponding text files.

Preprocessing of Live Scraped Data

1. **Image Validation**: The scraper checks images for validity, discarding those smaller than 100x100 pixels.
2. **Text Normalization**: Basic cleaning is performed on the text to remove URLs and special characters.
3. **Organization**: Cleaned text and valid images are prepared for upload to MongoDB, ensuring only high-quality data is stored for analysis.

5. Methodology

Image Classification:

Architecture: The image classification model is built on VGGNet, a convolutional neural network known for its depth and performance in image recognition tasks. Utilizing transfer learning, we leverage a pre-trained VGG19 model (trained on ImageNet) and fine-tune it for our specific disaster classification task. The top layers of the VGG19 model are removed, and custom dense layers are added to adapt it to our dataset.

Training: The training process involves several key steps:

- **Data Preparation**: Images are resized to 150x150 pixels and normalized. Data augmentation techniques, such as rotation, width and height shifts, shearing, zooming, and horizontal flips, are applied to increase dataset diversity and combat overfitting.
- **Hyperparameters**: We use a batch size of 32, the Adam optimizer, and categorical crossentropy loss function. The model is trained for a specified number of epochs (adjustable based on validation performance) with callbacks like early stopping and learning rate reduction to optimize training.
- **Callbacks**: Essential callbacks include ModelCheckpoint for saving the best model, EarlyStopping to prevent overfitting, and ReduceLROnPlateau for adaptive learning rate adjustment based on validation loss.

Evaluation Metrics: To assess model performance, the following metrics are utilized:

- **Accuracy**: The proportion of correctly classified instances.
- **Precision**: The ratio of true positive predictions to the total predicted positives.
- **Recall**: The ratio of true positive predictions to the total actual positives (also known as sensitivity).
- **F1-Score**: The harmonic mean of precision and recall, providing a single score that balances both.
- **Confusion Matrix**: A matrix that summarizes the performance of the classification model by comparing actual and predicted classes.
- **ROC Curve and AUC-ROC**: The Receiver Operating Characteristic curve plots true positive rate against false positive rate, and AUC-ROC quantifies the area under this curve, indicating the model's ability to discriminate between classes.

Text Classification

Models:
The project utilizes several machine learning models:

- **Logistic Regression**: A linear model for binary classification, effective for text data.
- **Random Forest**: An ensemble method that combines multiple decision trees for robust predictions.
- **XGBoost**: An optimized gradient boosting framework that improves accuracy and speed.
- **Support Vector Machine (SVM)**: A powerful model that works well for high-dimensional data.
- **Recurrent Neural Network (RNN)**: Specifically, a stacked LSTM model designed for sequence prediction tasks in text.

Preprocessing:
The text data undergoes the following preprocessing steps:

- **Tokenization**: Splitting text into individual words.
- **Stopword Removal**: Filtering out common words (e.g., "the", "and") while retaining specific pronouns to maintain context.
- **Lemmatization**: Reducing words to their base form to enhance feature consistency.

Feature Extraction:
Text features are extracted using:

- **TF-IDF (Term Frequency-Inverse Document Frequency)**: This method transforms text into numerical vectors reflecting the importance of words.
- **Embeddings**: Word embeddings are used in RNN models for capturing semantic meanings.

Evaluation:
The model performance is assessed using various metrics:

- **Accuracy**: The proportion of correct predictions.
- **Precision**: The ratio of true positives to the total predicted positives.
- **Recall**: The ratio of true positives to actual positives (sensitivity).
- **F1-Score**: The harmonic mean of precision and recall.
- **Confusion Matrix**: A visual representation of model predictions.
- **ROC Curve and AUC-ROC**: Measures of the model's ability to discriminate between classes.

Twitter Scraper:

The Twitter Scraper extracts text and images from tweets based on a specified hashtag, processes the data, and uploads it to MongoDB. Key components include:

1. **Setup**:
   - **Libraries**: Utilizes Selenium for web scraping, PIL for image processing, requests for downloading images, and pymongo/GridFS for MongoDB operations.
   - **WebDriver**: Initializes a Chrome WebDriver to navigate Twitter and scroll through tweets dynamically.
2. **Scraping Functionality**:
   - **Data Extraction**: Captures tweet text and images, saving text to files and downloading images to organized folders (Post_1, Post_2).
   - **Scrolling**: Continues to scroll until a set number of posts (max_posts) are collected.
3. **Cleaning Process**:
   - Validates downloaded images, removing those that are invalid or too small to ensure dataset quality.
4. **Uploading Process**:
   - **PostUploader**: Reads cleaned data, uploads images to MongoDB using GridFS, and stores tweets as documents. Local directories are cleared after uploads.
5. **Web Integration**:
   - Integrated into a Flask API, allowing users to trigger scraping with a hashtag and receive a success message after completion.

6. Web Application

The web application serves as the interface for users to classify images and texts, scrape Twitter for real-time data, and view predictions.

1. **Technologies**:
   - **Flask**: A lightweight Python web framework used to build the server-side application. It handles routing, requests, and rendering templates.
   - **Front-End Frameworks**: While the code snippet does not specify, frameworks like Bootstrap or React could enhance the UI/UX.
2. **Features**:
   - **Upload Tabs**: Separate tabs for image and text classification allow users to upload their data for analysis.
   - **Real-Time Scraping Interface**: Users can initiate the Twitter scraping process by providing a hashtag and a limit on the number of tweets to collect. This is done through the /start_scraper route, which utilizes the TwitterScraper class.
   - **Output Display**: Predictions (disaster or non-disaster) are returned in JSON format after classification. Users receive a success message along with the predicted results upon completion.
3. **Code Overview**

- **Flask Initialization**: The app is initialized with CORS enabled for cross-origin requests.
- **Routes**:
  - **Home**: Renders the about page.
  - **Classification Pages**: Separate routes for image and text classification interfaces.
  - **Live Scraper**: Route for accessing the live scraping feature.
  - **Scraping Functionality**: The /start_scraper endpoint triggers the scraping, cleaning, and uploading process.
  - **Prediction Endpoint**: The /show_prediction route runs predictions on scraped data and returns the results.
- **Image & Text Classification**: Routes handle file uploads and execute the classification models. The image classification route processes uploaded images, while the text classification route analyzes provided text.
- **Error Handling**: The application includes exception handling to manage and log errors effectively, providing informative responses to the user.

7. Results

Image Classification Results:

Overall Metrics-
Accuracy: 0.81 | Precision: 0.74 | Recall: 0.52 | F1-Score: 0.57 | Specificity: 0.85

Analysis of Results

1. **Performance Overview**: The model achieved **81% accuracy**, but precision and recall vary significantly across classes.
2. **Class-Specific Insights**:
   - **Class 4** performed best with **98% recall** and **91% F1-score**.
   - Classes **0**, **2**, and **3** struggled with low recall, indicating misclassification issues.
3. **Challenges**: Class imbalance affects performance, especially for underrepresented classes, and model complexity may lead to overfitting.

Future Directions

Implement data augmentation, explore class weighting, and analyze misclassifications to improve accuracy across all classes.

Text Classification Results:

1. **Logistic Regression:**
   Accuracy: 0.84 | Precision: 0.84 | Recall: 0.53 | F1-Score: 0.65 | Specificity: 0.96 | AUC-ROC: 0.74

   Analysis: High accuracy and precision indicate that the model effectively identifies non-disaster tweets. However, the lower recall suggests it misses some disaster-related tweets, affecting its sensitivity in critical situations.

2. **Random Forest:**
   Accuracy: 0.83 | Precision: 0.78 | Recall: 0.56 | F1-Score: 0.65 | Specificity: 0.94 | AUC-ROC: 0.75

   Analysis: Random Forest shows balanced performance but with slightly lower precision and recall compared to Logistic Regression. Its high specificity indicates good identification of non-disaster tweets, but it still misses many disaster tweets.

3. **XGBoost:**
   Accuracy: 0.82 | Precision: 0.80 | Recall: 0.45 | F1-Score: 0.58 | Specificity: 0.96 | AUC-ROC: 0.71

   Analysis: XGBoost demonstrates strong precision but a low recall, indicating it is conservative in predicting disasters. This model is good for specific non-disaster classifications but struggles to capture disaster events.

4. **SVM:**
   Accuracy: 0.82 | Precision: 0.81 | Recall: 0.70 | F1-Score: 0.68 | Specificity: 0.86 | AUC-ROC: 0.78

   Analysis: SVM exhibits a good balance between precision and recall, making it effective at identifying disaster-related tweets. Its performance in both precision and recall suggests it can adapt well to varying distributions of classes.

5. **RNN:**
   Accuracy: 0.79 | Precision: 0.63 | Recall: 0.61 | F1-Score: 0.62 | Specificity: 0.86 | AUC-ROC: 0.80

   Analysis: The RNN model shows lower precision, indicating a higher rate of false positives. While it maintains decent recall, its performance may not be ideal for high-stakes disaster detection.

6. **LSTM:**
   Accuracy: 0.81 | Precision: 0.65 | Recall: 0.68 | F1-Score: 0.66 | Specificity: 0.86 | AUC-ROC: 0.83

   Analysis: LSTM achieves a balance between precision and recall, but it still suffers from some misclassifications. The overall performance is acceptable, but improvements could enhance its reliability in disaster detection.

Best Model Selection

- **Best Model: SVM**
  Recall: 0.70 | Precision: 0.80 | F1-Score: 0.75 | AUC-ROC: 0.79
  Analysis: SVM emerges as the best choice for disaster text classification, offering a solid recall that helps capture more disaster-related tweets while maintaining good precision and overall balanced performance.

8. Discussion

Strengths and Weaknesses of the System

- **Strengths:**
  - **Robust Scraper:** Handles a large volume of posts reliably.
  - **Image and Text Integration:** Enhances disaster detection accuracy.
- **Weaknesses:**
  - **Lack of Language Check:** May result in irrelevant data being processed.
  - **Performance Limitations:** Slow scraping due to image loading times and local memory constraints limit post processing.
  - **Monolithic Architecture:** Simplicity hinders scalability and future enhancements.

Addressing Class Imbalance

- **Image Classification:** Merged fine classes into broader categories to create a more balanced dataset.
- **Text Classification:** Combined non-disaster text and assigned higher weights to disaster tweets to enhance sensitivity.

Importance of Combining Image and Text

- **Comprehensive Understanding:** Offers a richer context by leveraging visual and textual information.
- **Improved Decision Making:** Reduces false negatives, leading to more reliable disaster detection.
- **Real-Time Insights:** Enables quicker responses during critical situations.

Challenges with Real-Time Scraping

- **Rate Limits:** Social media APIs restrict data collection speed.
- **Data Noise:** Irrelevant posts complicate filtering and classification.
- **Processing Delays:** Slow image loading affects timely data gathering.

9. Conclusion

This project underscores the critical role of multi-modal classification (combining text and image data) in enhancing disaster detection and response. The integration of real-time Twitter scraping provides timely insights, which is essential for effective disaster management. The findings demonstrate that while significant progress has been made, challenges such as class imbalance and model performance still exist. Future enhancements could involve expanding the system to include more social media sources and improving model accuracy, making it a robust tool for disaster detection.

10. Future Work

1. **Enhancing Model Accuracy:** Further research into advanced deep learning techniques and hyperparameter tuning can improve classification performance.
2. **Use**
3. **Scaling the Scraper:** Optimizing the scraper to handle larger datasets and more frequent scraping will enhance real-time capabilities.
4. **Deploying in Real-World Environments:** Implementing the system in live scenarios with real-time data feeds can validate its effectiveness and reliability.
5. **Exploring Additional Features:** Investigating the use of geolocation data and sentiment analysis from tweets could further enrich disaster detection capabilities and context understanding.
6. **User Engagement:** Developing features for user feedback on classifications can help refine model performance and adapt to evolving language and trends in disaster-related communication.