Disaster Tweet Analyzer - Project Documentation

1. Introduction

Social media platforms, particularly Twitter, have become an essential tool for disseminating real-time information during disasters. The purpose of this project is to develop a machine learning model that can automatically classify whether a tweet is related to a disaster event. By identifying disaster-related tweets in real-time, authorities and organizations can better understand public sentiment, gather critical information, and respond more effectively.

This document outlines the progress of the Disaster Tweet Analyzer project over the past two weeks, focusing on data preprocessing, feature engineering, model selection, and evaluation.

2. Dataset and Methodology

2.1 Dataset

The dataset consists of a collection of Twitter posts with associated metadata like text, keywords, and location. Each tweet is labeled as either disaster-related or non-disaster-related. Key features of the dataset include:

**Text**: The main content of the tweet.

**Location**: The geographical location of the tweet, which may be incomplete or missing.

**Keyword**: Keywords related to the tweet’s content, often extracted from the tweet itself.

2.2 Methodology

Step 1: Data Preprocessing

**1. Handle Null Values:**

* Missing values in the `location` and `keyword` fields were imputed with the placeholder "unknown.
* Tweets with excessive missing data were discarded to avoid noise in the analysis.

**2. Text Cleaning**

* + Special characters, URLs, and emojis were removed using regular expressions.
  + Tweets were converted to lowercase to ensure uniformity.
  + Stopwords (e.g., "the," "and") were removed using NLTK to focus on meaningful content.

**3. Tokenization:**

* + Each tweet was broken into individual words (tokens) using spaCy's tokenizer.

**4. Stemming/Lemmatization (Optional):**

* + Stemming and lemmatization were applied to reduce words to their base forms, improving feature consistency (e.g., "running" → "run").

Step 2: Feature Engineering

**1. Text Vectorization**

* + Bag of Words (BoW): A sparse matrix was created, representing the frequency of each word across the dataset.
  + TF-IDF: This method was applied to weigh words based on their importance in a tweet versus the entire corpus.
  + Word Embeddings: Pre-trained embeddings such as Word2Vec and GloVe were experimented with to capture the semantic relationships between words.

**2. Keyword & Location Encoding:**

* Keywords were included as additional features, and One-Hot Encoding was applied to the location` field to incorporate geographical information into the model.

Step 3: Model Selection

**1. Baseline Models:**

* **Logistic Regression**: Simple yet effective for binary classification.
* **Naive Bayes**: Useful for text-based models due to its probabilistic nature.

**2. Advanced Models:**

**Random Forest**: A decision-tree-based ensemble method.

**Gradient Boosting (XGBoost, LightGBM):** For handling complex patterns and feature interactions.

**3. Deep Learning Models:**

* **LSTM/GRU**: Designed for handling sequential data, these models were tested for their ability to capture context in tweets.
* **Transformer-based Models (BERT, RoBERTa):** State-of-the-art models for text classification tasks, using Hugging Face's Transformer library.

3. Results

Model Performance

We evaluated several models on the preprocessed data. Below are the results from key model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-Score |
| Logistic Regression | 0.85 | 0.82 | 0.82 | 0.81 |
| Naive Bayes | 0.83 | 0.81 | 0.78 | 0.79 |
| Random Forest | 0.87 | 0.84 | 0.83 | 0.83 |
| XGBoost | 0.88 | 0.85 | 0.84 | 0.84 |
| BERT | 0.92 | 0.90 | 0.88 | 0.89 |

Evaluation Metrics:

* Accuracy: Overall correctness of the model.
* Precision: Proportion of correctly predicted disaster tweets out of all predicted disaster tweets.
* Recall: Proportion of correctly predicted disaster tweets out of all actual disaster tweets.
* F1-Score: Harmonic mean of precision and recall, useful for imbalanced datasets.

The BERT model, leveraging pre-trained transformers, showed the highest performance with an F1-score of 0.89.

4. Conclusion

This project aimed to build a text classification model to predict whether a tweet is related to a disaster. We successfully explored various preprocessing steps, feature engineering techniques, and modeling approaches. The best-performing model was the BERT-based transformer, which outperformed other models, achieving an accuracy of 92% and an F1-score of 0.89.

5. Future Objectives for the Next Two Weeks

* Model Optimization: Apply hyperparameter tuning using Grid Search or Random Search for models like BERT and XGBoost to further improve performance.
* Data Augmentation: Increase the size of the training data using techniques like data augmentation or collecting more labeled disaster-related tweets.
* Deployment: Begin the process of deploying the trained model for real-time or batch processing of Twitter posts.
* Model Interpretability: Investigate methods like SHAP or LIME to interpret the model’s predictions and understand feature importance.

6. References

* Hugging Face Transformers: https://huggingface.co/transformers/
* Scikit-learn Documentation: https://scikit-learn.org/
* Natural Language Toolkit (NLTK): https://www.nltk.org/
* TensorFlow: https://www.tensorflow.org/
* PyTorch: https://pytorch.org/
* GloVe Word Embeddings: https://nlp.stanford.edu/projects/glove/