MAIS 202
Deliverable 2
Team members: Ruolin Hu, Sihui Wei, Shengyi Zhong

Social Media Enabled Crisis Informatics

1. Problem statement:

We are building a crisis analysis platform using preprocessed tweets to extract key information about disasters. The outputs would include classified tweet feeds.

2. Data Preprocessing:

a. Datasets:

We are working with three datasets from Twitter Datasets from Crises [1], CrisisNLP: the Chile Earthquake (2014), California Earthquake (2014), and Pakistan Earthquake (2013) [1]. The datasets are provided in English TSV files containing 368,630, 254,525, and 156,905 tweets, respectively. These datasets were merged into a single pandas dataframe with the following structure: `Tweet_id`, `tweet_text`, and `label`. The dataset includes 9 labels, although there is a significant class imbalance:

- **other_useful_information**: 724 samples
- **injured_or_dead_people**: 326 samples
- **not_related_or_irrelevant**: 312 samples
- **donation_needs_or_offers_or_volunteering_services**: 299 samples
- **sympathy_and_emotional_support**: 106 samples
- **caution_and_advice**: 69 samples
- **infrastructure_and_utilities_damage**: 28 samples
- **displaced_people_and_evacuations**: 12 samples
- **missing_trapped_or_found_people**: 5 samples

b. Preprocessing:

We started by cleaning the data to remove noise and irrelevant information. We used NLTK's English stopword list to eliminate common words like "the", "is", and "at" that provide little value for classification. Mentions (e.g., @username) and URLs were removed as they do not contribute meaningful information for disaster classification and could introduce noise. Punctuation marks were stripped, except for hashtags, as they often contain useful categorical information in tweets. We applied the PorterStemmer to reduce words to their base forms, which helps group similar words together for classification. We used unigrams (single words) as features for our model.

After cleaning, the text was vectorized using TF-IDF (Term Frequency-Inverse Document Frequency), transforming the words into numerical features. This created a sparse matrix based on word frequency where TF (Term Frequency) measures how often a word appears in a tweet, and IDF (Inverse Document Frequency)

measures how unique or rare that word is across all tweets. This preprocessing prepared the data for classifier training using the generated TF-IDF vectors.

3. Machine Learning model:

a. Framework and tool:

For this preliminary deliverable, we focused on foundational work by training, testing, and comparing the performance of Random Forest and Naive Bayes algorithms. While our long-term goal is to implement a CNN for disaster tweet classification, we concentrated on these simpler models to establish a baseline at this stage.

For our preliminary implementation, we used the following libraries and tools:

- **scikit-learn** for both Random Forest and Naive Bayes implementations
- **NLTK** for text preprocessing
- **pandas** for data manipulation
- **numpy** for numerical operations

Since Random Forest and Naive Bayes are traditional machine learning models, they do not follow a layered architecture like neural networks (e.g., CNNs). Instead, Random Forest is an ensemble method consisting of multiple decision trees, while Naive Bayes is a probabilistic model based on Bayes' theorem.

b. Dataset splits, regularization/optimization techniques, and hyper-parameters:

We used an 80/20 train-test split for the dataset, with 10-fold cross-validation applied to the training data for a preliminary comparison between Random Forest and Naive Bayes. Random Forest outperformed Naive Bayes in terms of average accuracy. After obtaining preliminary accuracy for the two models using training and validation data, we proceeded with hyperparameter tuning.

To optimize the models' performance, we conducted a 10-fold grid search on the training data, varying the following hyperparameters:

- `n_estimators`: [100, 200, 300]
- `max_depth`: [10, 20, 30, None]
- `min_samples_split`: [2, 5, 10]
- `min_samples_leaf`: [1, 2, 4]

The optimal hyperparameters for Random Forest were:

- `n_estimators`: 100
- `max_depth`: None
- `min_samples_split`: 10
- `min_samples_leaf`: 1

For Naive Bayes, we performed a 10-fold grid search, varying the `alpha` hyperparameter:

- `alpha`: [0.1, 0.5, 1.0, 5.0, 10.0]

The optimal `alpha` for Naive Bayes was determined to be 0.5.

The Random Forest and Naive Bayes models with the best hyperparameters have been selected and stored. Finally, both models (Random Forest and Naive Bayes) were tested using the test data with their respective optimal hyperparameters.

c. Validation methods:

For Random Forest, the cross-validation accuracy from the initial runs, the accuracy after hyperparameter tuning, and the accuracy on the test set were all very similar, indicating no signs of overfitting. However, the accuracy hovered around 0.75, which suggests that the model may be slightly underfitting. Despite tuning, we did not observe any significant improvement in performance, which implies that further adjustments to the model or additional feature engineering might be necessary to achieve better results.

d. Challenges:

Our team encountered difficulties due to limited experience with text processing techniques, as we are not familiar with preprocessing steps such as tokenization, stopword removal, and stemming. We referred to related research papers to address this.

When dividing the dataset into folds for cross-validation, we also encountered substantial delays in computation time. By adjusting the number of folds from 10 to 5, we were able to substantially reduce the processing time while still maintaining reliable evaluation metrics.

4. Preliminary results:

- **10-Fold Cross-Validation**

1. Naive Bayes Classifier
   - Accuracy Scores:
     [0.7036, 0.6961, 0.6735, 0.6803, 0.6825, 0.6757, 0.7029, 0.7029, 0.6961, 0.6576]
   - Average Accuracy: 68.71%

2. Random Forest Classifier
   - Accuracy Scores:
     [0.7308, 0.7234, 0.7415, 0.7528, 0.7506, 0.7483, 0.7415, 0.7415, 0.7664, 0.7234]
   - Average Accuracy: 74.20%

Overall, the random forest model has a higher average accuracy of 74.20%.

- **Optimized Random Forest -** After performing hyperparameter tuning (reduced from 10-fold to 5-fold to save time)

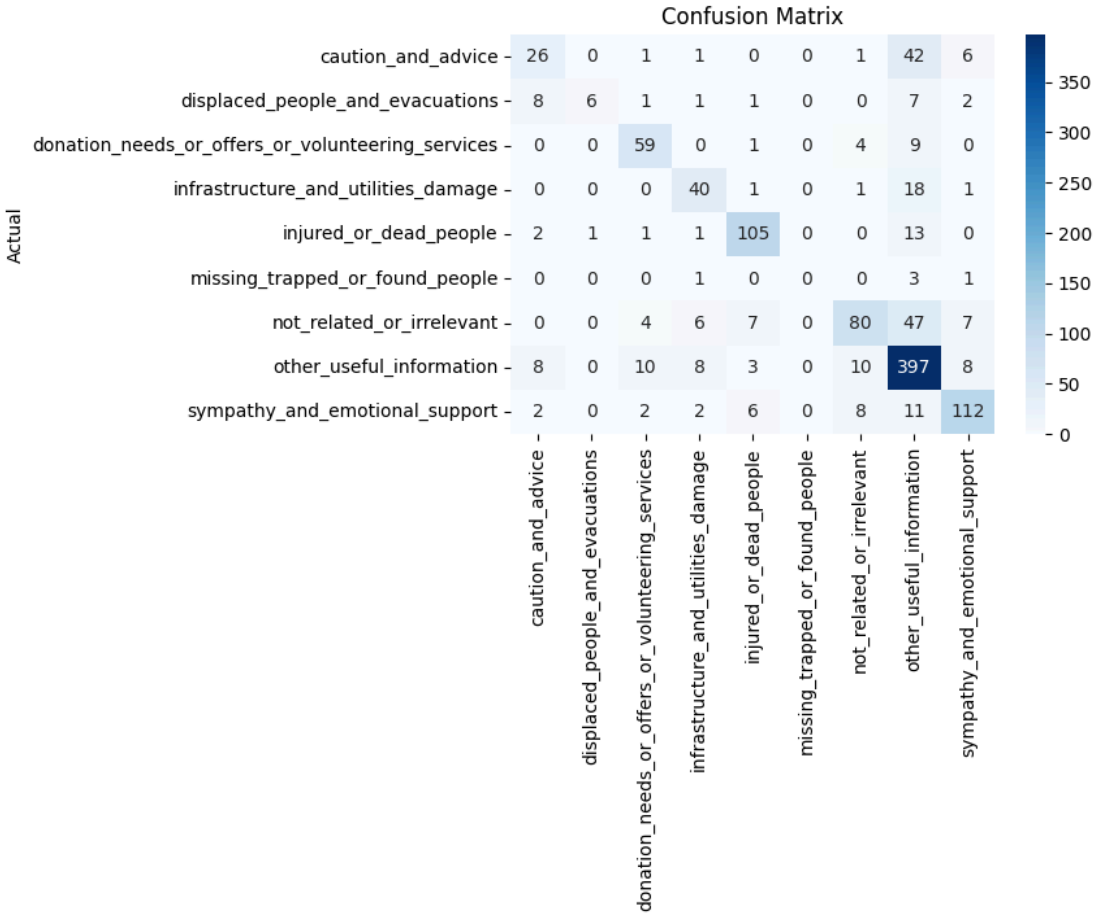
  1. Classification Report

```
Classification Report:
                                                precision    recall  f1-score   support

                                caution_and_advice       0.58      0.34      0.43        77
                      displaced_people_and_evacuations       0.88      0.27      0.41        26
donation_needs_or_offers_or_volunteering_services       0.76      0.84      0.80        73
                  infrastructure_and_utilities_damage       0.66      0.64      0.65        61
                                 injured_or_dead_people       0.84      0.85      0.85       123
                       missing_trapped_or_found_people       0.00      0.00      0.00         5
                            not_related_or_irrelevant       0.78      0.54      0.64       151
                              other_useful_information       0.73      0.90      0.81       444
                       sympathy_and_emotional_support       0.83      0.80      0.81       143

                                             accuracy                           0.75      1103
                                            macro avg       0.67      0.57      0.60      1103
                                         weighted avg       0.75      0.75      0.74      1103
```

  2. Confusion Matrix



The optimized Random Forest model with the identified best hyperparameters is performing well, achieving 75.04% accuracy during cross-validation.

The model is quite flexible (with no depth limit and fine-grained splits), but it may require further adjustments if overfitting or computational efficiency becomes a concern.

The model also struggles with underrepresented classes like missing_trapped_or_found_people and displaced_people_and_evacuations, with the results being misclassified across several other categories.

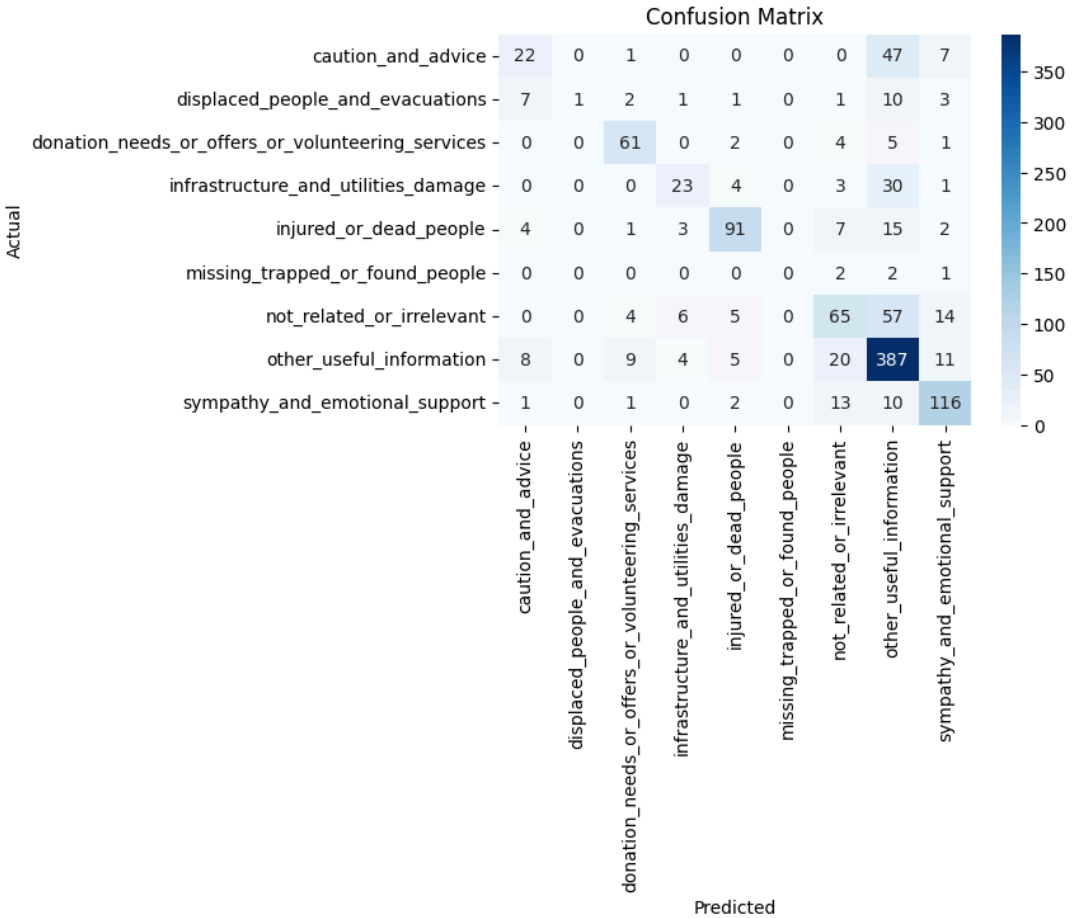
- **Optimized Naive Bayes -** After performing hyperparameter tuning
    1. Classification Report

```
Classification Report:
                                                precision    recall  f1-score   support

                                caution_and_advice       0.52      0.29      0.37        77
                        displaced_people_and_evacuations       1.00      0.04      0.07        26
donation_needs_or_offers_or_volunteering_services       0.77      0.84      0.80        73
                    infrastructure_and_utilities_damage       0.62      0.38      0.47        61
                                   injured_or_dead_people       0.83      0.74      0.78       123
                        missing_trapped_or_found_people       0.00      0.00      0.00         5
                                not_related_or_irrelevant       0.57      0.43      0.49       151
                             other_useful_information       0.69      0.87      0.77       444
                        sympathy_and_emotional_support       0.74      0.81      0.78       143

                                              accuracy                           0.69      1103
                                             macro avg       0.64      0.49      0.50      1103
                                          weighted avg       0.69      0.69      0.67      1103
```

    2. Confusion Matrix



The optimized Naive Bayes model still has a lower average accuracy (69%) than the random forest model.

5. Next steps:

To enhance our models' performance, we plan to explore more feature representations and techniques. We can potentially incorporate bigrams in our text preprocessing to capture contextual relationships between word pairs, which may enhance classification accuracy. Additionally, we may also implement information gain for feature selection prior to passing the vectors into TF-IDF, making use of only the most informative words.

We can also switch from TF-IDF to Word2Vec for feature representation. While TF-IDF is effective in transforming text into numerical features, it does not create word embeddings that account for the semantic relationships between words (and the effects of bigrams are limited).

We also plan to implement Convolutional Neural Networks (CNNs), as they automatically learn features from raw text data, eliminating the need for manual feature engineering [2]. We plan on using Word2Vec or BERT for embedding representations within the CNN architecture.

References

[1] Imran, M., Mitra, P., & Castillo, C. (2016). Twitter as a lifeline: Human-annotated Twitter corpora for NLP of crisis-related messages. *In Proceedings of the 10th Language Resources and Evaluation Conference (LREC)* (pp. 1638–1643). Portorož, Slovenia.

[2] Nguyen, D. T., Al-Mannai, K. A., Joty, S., Sajjad, H., Imran, M., & Mitra, P. (2017). Robust classification of crisis-related data on social networks using convolutional neural networks. *In Proceedings of the 11th International AAAI Conference on Web and Social Media (ICWSM)*. Montreal, Canada.