NLP with Disaster Tweets

YUNG-TAI TAI
Computer Science and Business Information Systems
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
tayu1011@h-ka.de

JOU-CHEN LIU
Computer Science and Business Information Systems
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
lijo1026@h-ka.de

YI-JING LIAN
Computer Science and Business Information Systems
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
liyi1012@h-ka.de

***Abstract*—The main objective of this study is to find a pattern where in Tweets can be properly sorted out into those that talk about real accidents or do not. Four approaches were applied to build the future predicting methods: XGBoost with TF-IDF, LSTM with Word2Vec embeddings, GRU with Word2Vec embeddings, and DistilBERT. A comparison of these models is also provided.**

***Keywords*—NLP, Tweet classification, disaster, XGBoost, TF-IDF, LSTM, Word2Vec, GRU, DistilBERT**

I. INTRODUCTION

The efficient identification of disaster-related Tweets is a vital necessity for the effectiveness of disaster response and management. In the first part of this paper, we consider four different methods to determine whether a Tweet is a real disaster or not. The models assessed are:

1. XGBoost with TF-IDF
2. LSTM with Word2Vec embeddings
3. GRU with Word2Vec embeddings
4. DistilBERT

The main aim of this study is to find the most effective model out of these four. The paper is organized as follows: Section II represents the problem statement, Section III describes the methodology, Section IV presents the evaluation and comparison of the models, and Section V concludes the study.

II. PROBLEM STATEMENT

Add to that the incredibly important role of social media platforms, the fact that we always depend on being informed about what's going on in disasters. While this is the case, misleading or irrelevant information is often part of Tweets. The classification of disaster-related Tweets data is challenging because of the lack of brevity, informal communication, and contextual nuances that Tweets have with non-disaster-related ones. One of the major issues is that the noisy and unstructured nature of Tweets is difficult to handle.

The main challenges are:

- Removing the noise and structuring the Tweets.
- Guaranteeing that the semantic context is identified accurately.
- Running the model correctly but also being efficient on the computer.

The study is dedicated to finding these challenges, through the use of machine learning and deep learning techniques that are diverse and now are to be found in the next section.

Regarding the dataset, we obtained it from the Kaggle platform[1]. The dataset includes four columns: keyword, location, text, and target. Our goal is to use "keyword, location, and text" as features to predict the "target," where a value of "1" means there is a real disaster, and "0" means there is no disaster. We divided the train.csv file into 80% for training and 20% for testing, since the test.csv file does not contain ground truth labels.

III. APPROACH

The four approaches employed in this study are as follows:

A. XGBoost x TF-IDF — Baseline method

Term Frequency-Inverse Document Frequency (TF-IDF) is the most common way of text representation, basically used to measure the significance of a given word in a document, and it is extensively used in information retrieval and text mining such as the hallmark of its conceptual framework. The core functions and roles of TF-IDF include: measuring the importance of a word, the text vector representation of transfer, and the filtering out of noise. TF-IDF is an age-old text representation method that can numerically evaluate the words' significance in a document, an armory that enables you to perform tasks like classification based on text, information retrieval, and topic analysis amongst others: it is a very useful tool for all of these.

eXtreme Gradient Boosting(XGBoost) is a very strong gradient boosting algorithm framework that is most widely encountered in the context of machine learning tasks, especially for classification and regression with the use of structured data (e.g. tabular data). Indeed, it is the functionality and precision that makes XGBoost that famous.The benefits of XGBoost are: the fast implementation, the high accurateness and the function that allows you to select the suitable objective (e.g. binary classification, multivariate classification, regression).

In the text categorization task, the combination of TF-IDF and XGBoost fully utilizes the advantages of both, which is an efficient and accurate solution. Here are the applications and benefits of the combination:

- Smooth integration of numericization and classification:
  - After TF-IDF numericizes text features, it can be directly used as input for XGBoost without additional conversion, which greatly simplifies the workflow of text classification.

- Noise immunity:
    - TF-IDF has already reduced the influence of high-frequency useless words through the IDF mechanism.
- XGBoost itself is also insensitive to the influence of noise features through the characteristics of the tree model, which further improves the stability of the model.
- High efficiency and accuracy:
    - Combining the efficient feature extraction of TF-IDF and the powerful classification capability of XGBoost, we can quickly obtain highly accurate classification results.

The combination of TF-IDF and XGBoost provides a fast, accurate, and explanatory text categorization solution. In this disaster tweet classification task, they are responsible for numericization of text information and efficient learning of data schema respectively, which work in tandem to achieve high-performance of the text classification task.

Due to the nature of TF-IDF itself, we don't need to delete the stopwords in the text, so we only need to deal with the subsequent parameter optimization to achieve better performance.

Since this method is a simple machine learning application, it is also our baseline model.

B. LSTM with word2Vec

Long Short-Term Memory (LSTM) networks leverage sequential dependencies within text data, making them effective for capturing context over time. Word2Vec embeddings are employed to convert words into dense vectors that encapsulate semantic meanings.

During the data preprocessing step, we performed the following actions:

- Removed URLs, HTML tags, emojis, and punctuation.
- Tokenized the text column using the NLTK library.
- Removed stop words from the text column.

Subsequently, we built a Word2Vec model and used it to construct a vocabulary list and a word-to-index mapping. The text column was then converted into indices based on this mapping. The processed data was split into training and testing datasets.

To address overfitting observed in our LSTM model, we incorporated dropout layers and implemented an early stopping mechanism to optimize the training process.

C. GRU with word2Vec

Gated Recurrent Unit (GRU) networks are a simplified alternative to LSTMs, designed to reduce computational complexity while maintaining comparable performance. Like in the LSTM approach, Word2Vec embeddings are used to represent words as dense semantic vectors.

The preprocessing and model preparation steps for GRU were similar to those used for the LSTM model. These steps included cleaning the text data, tokenizing it, building the Word2Vec model, converting text to indices, and splitting the data into training and testing sets.

D. DistilBERT

DistilBERT is a distilled version of Bidirectional Encoder Representations from Transformers (BERT), a bidirectional language model that understands the meaning of words from their context and is widely used in natural language processing tasks.

First, is the data preprocessing step, we performed the following actions:

- Removed URLs, HTML tags, emojis, and punctuation.
- Tokenized the text column using the pre-trained DistilBERT model through AutoTokenizer.
- Truncated text that exceeds the maximum input length of the model.
- Used DataCollatorWithPadding to dynamically pad the data to ensure that the samples in the batch have the same length.

Next, is the model construction step, we loaded the pre-trained model based on DistilBERT via AutoModelForSequenceClassification and set as a binary classification task.

Last, is the model training. Mainly to prevent overfitting, we set an EarlyStopping mechanism to stop training when performance plateaued. For the training, we also defined training hyperparameters, optimizer, scheduler and evaluation metrics.

IV. EVALUATION

A. XGBoost x TF-IDF

The results of the TF-IDF with XGBoost model's performance are shown on Fig. 1 and Table 1.

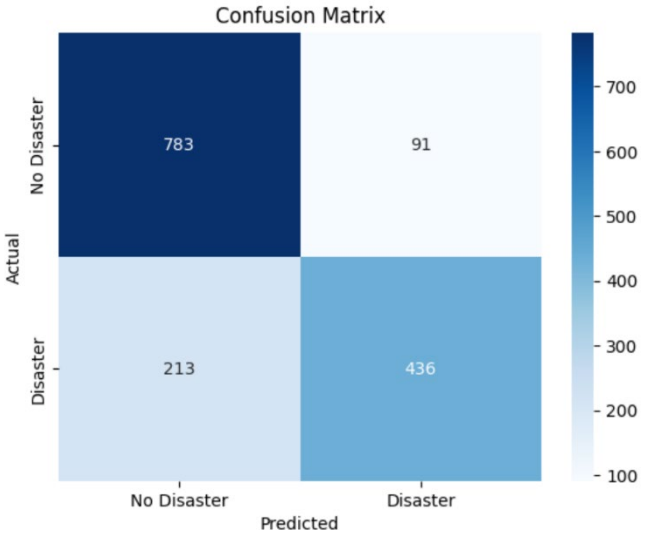

Fig. 1. Confusion matrix of TF-IDF with XGBoost

Table 1. Performance Metrics of TF-IDF with XGBoost

XGBoost x TF-IDF	precision	recall	**f1-score**	support
0	0.79	0.90	0.84	874
1	0.83	0.67	0.74	649
accuracy	-	-	0.79	1523
macro avg	0.81	0.78	0.79	1523
weighted avg	0.80	0.80	0.80	1523

- **Precision** measures the proportion of correctly predicted positive observations to the total predicted positive observations. The precision for class 0 is lower (0.79) than for class 1 (0.83), indicating the model is better at identifying class 1.
- **Recall** measures the proportion of correctly predicted positive observations to all actual positive observations. The recall for class 0 is higher (0.90) than for class 1 (0.67), indicating the model is better at identifying class 0.
- **F1-Score** is the harmonic mean of precision and recall, balancing their trade-off. The F1-scores for class 0 (0.84) and class 1 (0.74) reflect overall model effectiveness for each class.
- **Support** represents the number of actual occurrences of each class in the dataset. There are 874 instances of class 0 and 649 instances of class 1.
- **Accuracy** of 0.79 indicates that the model correctly predicted 79% of the total samples.
- **Macro average** calculates the unweighted mean of the metrics across all classes, treating each class equally. Precision: 0.81, Recall: 0.78, F1-Score: 0.79, showing that the model's performance is balanced across classes.
- **Weighted Average** considers the sample size of each class when calculating metrics. Precision: 0.80, Recall: 0.80, F1-Score: 0.80, representing the overall model performance across the entire dataset.

B. LSTM with word2Vec

The results of the LSTM model's performance are shown on Fig. 2 and Table 2.

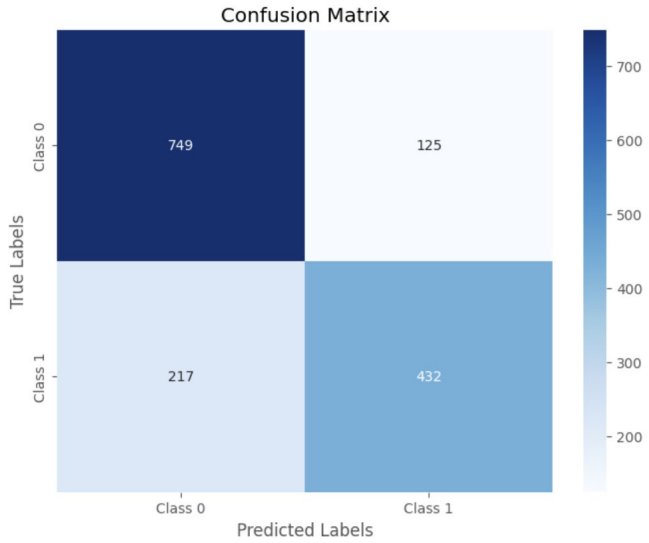

Fig. 2. Confusion matrix of LSTM

Table 2. Performance Metrics of LSTM

LSTM	precision	recall	**f1-score**	support
0	0.78	0.86	0.81	874
1	0.78	0.67	0.72	649
accuracy	-	-	0.78	1523
macro avg	0.78	0.76	0.77	1523
weighted avg	0.78	0.78	0.77	1523

- **Precision** measures the proportion of correctly predicted positive observations to the total predicted positive observations. For class 0 and class 1, the precision is 0.78 each, indicating a balanced ability to minimize false positives.
- **Recall** (or sensitivity) measures the proportion of correctly predicted positive observations to all actual positive observations. The recall for class 0 is higher (0.86) than for class 1 (0.67), indicating the model is better at identifying class 0.
- **F1-Score** is the harmonic mean of precision and recall, balancing their trade-off. The F1-scores for class 0 (0.81) and class 1 (0.72) reflect overall model effectiveness for each class.
- **Support** represents the number of actual occurrences of each class in the dataset. There are 874 instances of class 0 and 649 instances of class 1.
- **Accuracy** of 0.78 indicates that the model correctly predicted 78% of the total samples.
- **Macro average** calculates the unweighted mean of the metrics across all classes, treating each class equally. Precision: 0.78, Recall: 0.76, F1-Score: 0.77, showing that the model's performance is balanced across classes.
- **Weighted Average** considers the sample size of each class when calculating metrics. Precision: 0.78, Recall: 0.78, F1-Score: 0.77, representing the overall model performance across the entire dataset.

C. GRU with word2Vec

The results of the GRU model's performance are shown on Fig. 3 and Table 3.

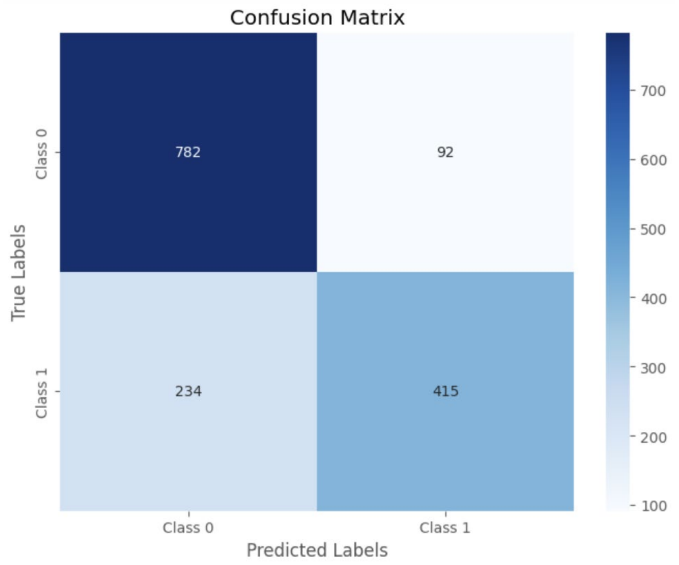

Fig. 3. Confusion Matrix of GRU

Table 3. Performance Metrics of GRU

GRU	precision	recall	**f1-score**	support
0	0.77	0.89	0.83	874
1	0.82	0.64	0.72	649
accuracy	-	-	0.79	1523
macro avg	0.79	0.77	0.77	1523
weighted avg	0.79	0.79	0.78	1523

- **Precision** measures the proportion of correctly predicted positive observations to the total predicted positive observations. For class 0, the precision is 0.77, and for class 1, it is 0.82, showing a slightly better performance in minimizing false positives for class 1.
- **Recall** (or sensitivity) measures the proportion of correctly predicted positive observations to all actual positive observations. The recall for class 0 is 0.89, while for class 1, it is 0.64, indicating that the model is significantly better at identifying class 0.
- **F1-Score** is the harmonic mean of precision and recall, balancing their trade-offs. The F1-score for class 0 is 0.83, and for class 1, it is 0.72, reflecting a stronger overall performance for class 0.
- **Support** represents the number of actual occurrences of each class in the dataset. There are 874 instances of class 0 and 649 instances of class 1.
- **Accuracy** of 0.79 indicates that the model correctly predicted 79% of the total samples.
- **Macro average** calculates the unweighted mean of the metrics across all classes, treating each class equally. Precision: 0.79, Recall: 0.77, F1-Score: 0.77, showing that the model's performance is balanced across classes.
- **Weighted Average** considers the sample size of each class when calculating metrics. Precision: 0.79, Recall: 0.79, F1-Score: 0.78, representing the overall model performance across the entire dataset.

D. DistilBERT

The results of the DistilBERT model's performance are shown on Fig. 4 and Table 4. In this study, DistilBERT achieved 4% higher accuracy than the baseline method—XGBoost x TF-IDF.

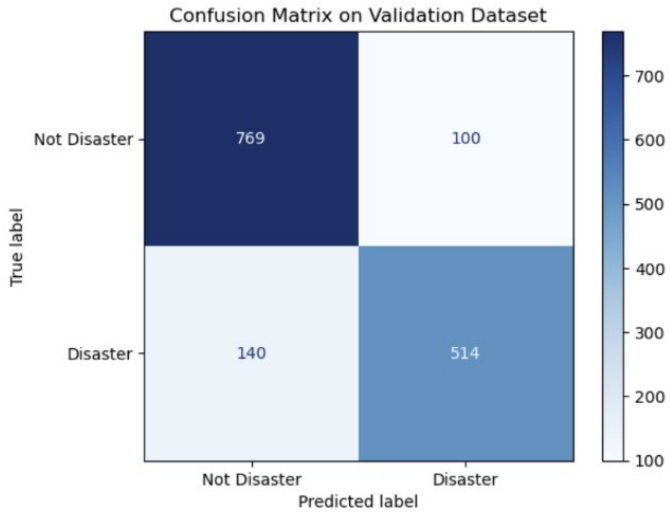

Fig. 4. Confusion matrix of DistilBERT

Table 4. Performance Metrics of DistilBERT

DistilBERT	precision	recall	**f1-score**	support
0	0.84	0.88	0.86	869
1	0.83	0.78	0.81	654
accuracy	-	-	0.84	1523
macro avg	0.84	0.83	0.83	1523
weighted avg	0.84	0.84	0.84	1523

- **Precision** measures the proportion of correctly predicted positive observations to the total predicted positive observations. For class 0, the precision is 0.84, and for class 1, it is 0.83, showing a strong and balanced ability to minimize false positives for both classes.
- **Recall** (or sensitivity) measures the proportion of correctly predicted positive observations to all actual positive observations. The recall for class 0 is higher (0.88) compared to class 1 (0.78), indicating that the model is better at identifying instances of class 0.
- **F1-Score** is the harmonic mean of precision and recall, balancing their trade-off. The F1-scores for class 0 (0.86) and class 1 (0.81) demonstrate the model's effectiveness for each class, with slightly better performance for class 0.
- **Support** represents the number of actual occurrences of each class in the dataset. There are 869 instances of class 0 and 654 instances of class 1, indicating a moderate class imbalance.

- **Accuracy** of 0.84 shows that the model correctly predicted 84% of the total samples, indicating a robust overall performance.
- **Macro Average** calculates the unweighted mean of the metrics across all classes, treating each class equally. Precision: 0.84, Recall: 0.83, F1-Score: 0.83, showing that the model performs well across both classes without favoring one over the other.
- **Weighted Average** considers the sample size of each class when calculating metrics. Precision: 0.84, Recall: 0.84, F1-Score: 0.84, reflecting the model's overall performance while accounting for the class imbalance in the dataset.

V. CONCLUSION

In this study, we evaluated four methods for disaster-related tweet classification: TF-IDF with XGBoost, LSTM with Word2Vec, GRU with Word2Vec, and DistilBERT. Each method leverages distinct strengths, offering insights into the effectiveness of traditional and modern NLP techniques for this task.

DistilBERT achieved the best performance with an accuracy of 84%, demonstrating its superiority in capturing global context and semantic relationships using the Transformer mechanism. This result highlights the power of pre-trained Transformer models, which excel in understanding complex language structures and nuanced contexts, even in short and sparse texts like tweets.

TF-IDF with XGBoost attained an accuracy of 80%, showcasing its efficiency as a lightweight, interpretable, and computationally inexpensive solution. Although it does not account for contextual relationships between words, its combination of TF-IDF's strong feature extraction and XGBoost's robust classification ability makes it a solid baseline for text classification tasks.

LSTM and GRU with Word2Vec yielded accuracies of 77% and 78%, respectively. These recurrent neural networks (RNNs) demonstrated their capacity to model sequential dependencies and long-term contextual relationships in text. However, their performance was slightly lower than TF-IDF with XGBoost, likely due to the limited dataset size and the challenges of training deep learning models from scratch without extensive pretraining.

REFERENCES

[1] "Natural Language Processing with Disaster Tweets," Kaggle. Available: https://www.kaggle.com/competitions/nlp-getting-started

[2] "Kaggle 灾难推文的自然语言处理-最佳得分详解-CSDN 博客." Available: https://blog.csdn.net/StrawBerryTreea/article/details/131948632

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, Nov. 1997.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in Proc. Int. Conf. Learning Representations (ICLR), 2013, pp. 1-12.

[5] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1724-1734.

[6] "DistilBERT." Available: https://huggingface.co/docs/transformers/model_doc/distilbert

[7] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," arXiv.org, Oct. 02, 2019. https://arxiv.org/abs/1910.01108

[8] "Text classification." Available: https://huggingface.co/docs/transformers/en/tasks/sequence_classification

[9] "early stopping in PyTorch," Stack Overflow. Available: https://stackoverflow.com/questions/71998978/early-stopping-in-pytorch