Disaster Tweet Detection Using NLP

Group 5: Rueiyu (Michelle) Chang, Catherine Chen, Jonah Kim, Lyla Liu, Esha Sidhu

ABSTRACT

Social media platforms have become crucial channels for real-time communication during emergencies, particularly Twitter. However, the prevalence of misinformation and the use of disaster-related terms in non-disaster contexts pose significant challenges for automated systems attempting to identify genuine disaster events. This project focuses on developing a natural language processing (NLP) model that can accurately classify disaster-related tweets using a dataset from Kaggle. We implemented and compared three models: Logistic Regression, Support Vector Machines (SVM), and Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) units. Our RNN model demonstrated superior performance, achieving a validation accuracy of 83.8%, due to its ability to capture sequential data and complex patterns.

By leveraging the RNN model, we aim to enhance the reliability of information disseminated during emergencies, which is crucial for timely and effective emergency responses. The performance of the model was thoroughly evaluated using various metrics, including validation accuracy, confusion matrix analysis, precision, recall, and F1-score. Our findings indicate that the RNN model significantly minimizes false positives and negatives, ensuring the quality and accuracy of disaster-related information. This, in turn, aids disaster response teams and news agencies in filtering out relevant information promptly and accurately.

Overall, this project underscores the importance of advanced NLP techniques in disaster tweet detection and highlights the potential for further improvements. Future work will focus on enhancing the model's robustness and generalization by exploring advanced architectures and larger datasets. The ultimate goal is to develop a real-time system capable of providing reliable and timely information during disaster events, thereby supporting effective disaster management and response efforts.

INTRODUCTION

With the widespread use of smartphones, many users share real-time information on platforms like Twitter, especially during emergencies. These posts can be crucial for disaster reports, providing timely updates on events such as natural disasters, accidents, and other emergencies. However, it can be challenging for machines to distinguish between genuine emergencies and false alarms. The ambiguity in language and the potential for misinformation complicate the task of accurately identifying disaster-related content. This complexity makes it a critical task for machine learning models to reliably discern true disaster events from misleading or non-relevant posts.

Our goal is to develop a natural language processing (NLP) model that can accurately classify disaster-related tweets. Misclassification of disaster-related tweets can lead to unnecessary panic and the possible waste of resources, as false alarms may trigger unwarranted emergency responses. Conversely, failing to identify genuine disaster-related tweets can result in delayed or inadequate responses to actual emergencies, potentially exacerbating the impact of the disaster.

By improving the accuracy of disaster-related tweet classification, we aim to enhance the detection of valid disaster warnings, thereby assisting in emergency response and mitigating the spread of misinformation. Accurate classification models can provide disaster response teams and news agencies with reliable information, enabling them to act swiftly and appropriately. This not only improves the efficiency and effectiveness of emergency responses but also helps in maintaining public trust and preventing the dissemination of false information.

To achieve this, we implemented and compared three machine learning models: Logistic Regression, Support Vector Machines (SVM), and Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) units. By leveraging the strengths of these models, particularly the RNN's ability to capture sequential and contextual information in tweets, we aim to develop a robust and reliable system for disaster tweet detection. The subsequent sections of this report detail the problem definition, data preparation, methods, experiments, results, and conclusions of our study, highlighting the potential and challenges of using NLP for disaster-related tweet classification.

PROBLEM DEFINITION & FORMALIZATION

Twitter has become a crucial platform for disseminating real-time information during emergencies. However, the content posted on Twitter is not always straightforward. Tweets may use figurative language, metaphors, or casual mentions of disaster-related terms without referring to actual events. The primary challenge is to accurately distinguish between tweets that are genuinely reporting disasters and those that are not. This problem is of significant importance as accurate classification can aid disaster response teams and news agencies in filtering out relevant information promptly.

The problem we aim to solve is binary classification of tweets. Specifically, we need to classify whether a tweet is related to a real disaster (label 1) or not (label 0). The input to our system is the text of a tweet, and the output is a binary label indicating the tweet's relevance to a disaster. This task involves several challenges, including the ambiguity in language, where tweets often contain slang, abbreviations, and ambiguous language that complicates understanding the context correctly. Furthermore, understanding the context in which words are used is crucial, as words like "ablaze" could refer to a literal fire or be used metaphorically to describe a beautiful sunset. Additionally, the

dataset contains only 7,614 tweets, which is relatively modest for training deep learning models. This limitation necessitates careful handling to prevent overfitting and ensure the model generalizes well to unseen data.

We formalize the problem as follows: given a dataset of tweets, each tweet being a sequence of words, the dataset includes the tweet's text and a binary label indicating whether the tweet is disaster-related. The goal is to develop a model that maximizes the accuracy of this classification task while also maintaining high precision and recall, particularly for the disaster-related class.

To evaluate the performance of the model, we use metrics such as accuracy, precision, recall, and F1-score. Accuracy measures the proportion of correctly classified tweets, while precision assesses the proportion of tweets classified as disasters that are actually disasters. Recall evaluates the proportion of actual disaster tweets that are correctly classified, and the F1-score, as the harmonic mean of precision and recall, provides a single measure of the model's accuracy, particularly useful when the class distribution is imbalanced.

Given the nature of the problem, we need a model that can handle the sequential and contextual nature of text data. While traditional models like Logistic Regression and SVM can provide baseline performance, they may struggle with the complexities of language. Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, are well-suited for this task as they can capture long-term dependencies and context within the tweet sequence.

In summary, the problem of disaster tweet detection is formalized as a binary classification task, with the goal of accurately distinguishing between disaster-related and non-disaster-related tweets. The challenge lies in the ambiguous nature of language and the modest size of the dataset. By leveraging advanced NLP techniques and suitable models, we aim to achieve high performance in this critical task.

DATA PREPARATION & PREPROCESSING

Effective data preparation and preprocessing are critical steps in any machine learning project, particularly in natural language processing (NLP) tasks involving textual data. For our project on disaster tweet detection, we employed several preprocessing steps to clean, transform, and prepare the data for model training and evaluation.

The dataset used in this project was obtained from a Kaggle competition titled "Natural Language Processing with Disaster Tweets." It consists of 7,614 tweets, each labeled as either disaster-related (1) or non-disaster-related (0). The dataset also includes additional metadata such as keywords and locations, which were not used in this study.

The first step in our data preparation pipeline was data cleaning. We focused on removing unnecessary columns and handling missing values. Specifically, we dropped the 'keyword,' 'location,' and 'id' columns from the dataset. These columns were deemed irrelevant for our text classification task. The 'keyword' and 'location' columns contained numerous missing values, and preliminary analyses indicated that they did not significantly contribute to the classification performance.

Text preprocessing is crucial in NLP as it transforms raw text into a format that can be effectively processed by machine learning models. Our text preprocessing pipeline included the following steps:

1. **Tokenization:** We used the Spacy tokenizer to split the tweet text into individual words or tokens. Tokenization helps break down the text into manageable pieces and is the first step in converting text into numerical data.
2. **Lowercasing:** All characters in the tweets were converted to lowercase. This step ensures uniformity and helps reduce the vocabulary size by treating 'Disaster' and 'disaster' as the same word.
3. **Removing Punctuation and Special Characters:** Punctuation marks and special characters were removed from the text. These characters often do not contribute to the semantic meaning and can introduce noise into the data.
4. **Stop Words Removal:** Commonly used words that do not carry significant meaning, known as stop words (e.g., 'the', 'and', 'is'), were removed from the tweets. Removing stop words helps focus on the words that contribute most to the classification task.
5. **Stemming and Lemmatization:** We applied lemmatization to reduce words to their base or root form. For example, 'running' becomes 'run.' Lemmatization helps in reducing the vocabulary size and capturing the core meaning of words.

After text preprocessing, the next step was to build a vocabulary from the training data. A vocabulary is a mapping of unique words to numerical indices. We used the torchtext library to create a vocabulary of the top 10,000 most frequent words in the training dataset. Words that appeared less frequently were replaced with a special '<unk>' token, representing unknown words.

With the vocabulary in place, we converted each tweet into a sequence of numerical indices corresponding to the words in the vocabulary. This step transforms the textual data into a format suitable for input into machine learning models. Each word in the tweet was replaced by its corresponding index from the vocabulary.

To ensure uniform input length for the models, we applied padding and truncation to the sequences. Tweets of varying lengths can pose challenges for batch processing in machine learning models. We padded shorter tweets with zeros and truncated longer tweets to a fixed length, ensuring that all input sequences had the same length. This fixed length was chosen based on the distribution of tweet lengths in the dataset.

Finally, we split the dataset into training and validation sets. We used 80% of the data for training and 20% for validation. This

split allowed us to train the models on a substantial portion of the data while reserving a significant portion for evaluating the models' performance on unseen data.

METHOD DESCRIPTION

In this project, we implemented three distinct machine learning models: Logistic Regression, Support Vector Machines (SVM), and Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM). Each model was selected based on its potential to handle the unique challenges posed by the dataset, specifically the need to classify tweets accurately as either disaster-related or non-disaster-related. Below, we provide a detailed description of the steps involved in implementing each model. Our code is linked here.

1 Logistic Regression and SVM

Logistic Regression and SVM are traditional machine learning models often used as baselines in classification tasks. The first step in implementing these models involved vectorizing the text data. We used the Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer from the scikit-learn library to convert the tweets into numerical representations. TF-IDF is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents. This conversion is crucial as it transforms the raw text data into a format that the models can process.

Once the text data was vectorized, we split the dataset into training and validation sets, ensuring that 80% of the data was used for training and 20% for validation. For Logistic Regression, we utilized the logistic regression model from scikit-learn, applying L2 regularization to prevent overfitting. We trained the model by fitting it to the training data and used cross-validation to tune the hyperparameters, particularly the regularization strength.

For SVM, we implemented the model using scikit-learn's SVM classifier. The key step here was selecting the appropriate kernel function to handle non-linear relationships within the data. After experimenting with linear, polynomial, and radial basis function (RBF) kernels, we found that the RBF kernel provided the best performance. Like with Logistic Regression, we employed cross-validation to fine-tune the hyperparameters, focusing on the regularization parameter (C) and the kernel coefficient (gamma).

2 Recurrent Neural Networks (RNN - LSTM)

Given the sequential nature of tweet text, Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) units were a natural choice for this task. We implemented the RNN model using PyTorch, a popular deep learning library. The implementation involved several key steps:

1. **Embedding Layer:** The first layer in our RNN model was the embedding layer, which converted words into dense vector representations. We built a vocabulary from the training data and used this vocabulary to map each word to a corresponding vector. The embedding layer helps capture the semantic meaning of words and reduces the dimensionality of the input data.
2. **LSTM Layer:** The core of our model was a bidirectional LSTM layer. LSTMs are a type of RNN capable of learning long-term dependencies, making them suitable for handling sequences of text. The bidirectional configuration allowed the model to consider both past and future contexts in the tweet sequence. This layer processed the embedded word vectors and produced hidden states that captured the sequential information in the tweets.
3. **Fully Connected Layer:** Following the LSTM layer, we added a fully connected (dense) layer that performed the final classification. This layer received the hidden states from the LSTM and produced output logits for each class (disaster and non-disaster).
4. **Training Setup:** We trained the RNN model using the Adam optimizer, which is well-suited for training deep learning models due to its adaptive learning rate. The loss function used was CrossEntropyLoss, which is appropriate for binary classification tasks. We trained the model for 30 epochs, with a batch size of 32 for training and 128 for validation. To prevent overfitting, we applied a dropout rate of 0.6 to the LSTM layer.
5. **Evaluation:** During training, we monitored the model's performance on the validation set. We used accuracy, precision, recall, and F1-score as evaluation metrics. The model's predictions were compared to the actual labels, and the metrics were computed to assess the model's effectiveness.

In summary, our methodology involved carefully preprocessing the text data, selecting and tuning appropriate models, and employing advanced techniques like LSTM to capture the sequential nature of tweets. Logistic Regression and SVM served as strong baselines, while the RNN (LSTM) model leveraged the power of deep learning to handle complex patterns in the text data, ultimately providing superior performance in disaster tweet detection.

EXPERIMENTS DESIGN & EVALUATION

To evaluate the performance of the Logistic Regression, Support Vector Machines (SVM), and Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) models, we designed a series of experiments. The dataset was split into training and validation sets, with 80% of the data used for training and 20% for validation. This split ensured that the models were trained on a substantial portion of the data while allowing us to evaluate their performance on unseen data.

1 Training and Hyperparameter Tuning

For Logistic Regression and SVM, we used the scikit-learn library. The text data was first vectorized using the Term Frequency-Inverse Document Frequency (TF-IDF) method. We then trained the models and performed hyperparameter tuning

using cross-validation. For Logistic Regression, we focused on the regularization parameter, while for SVM, we experimented with different kernels (linear, polynomial, and RBF) and tuned the regularization parameter (C) and the kernel coefficient (gamma). The RBF kernel provided the best performance for SVM.

The RNN model was implemented using PyTorch. The tweets were tokenized and converted into sequences of numerical indices based on a vocabulary built from the training data. We used an embedding layer to convert these indices into dense vectors, followed by a bidirectional LSTM layer to capture sequential dependencies. The final dense layer performed the classification. The model was trained using the Adam optimizer and CrossEntropyLoss. We trained the model for 30 epochs with a batch size of 32 for training and 128 for validation. Dropout with a rate of 0.6 was applied to the LSTM layer to prevent overfitting.

2 Evaluation Metrics

We evaluated the models using several metrics: accuracy, precision, recall, and F1-score. These metrics provide a comprehensive view of the model's performance, particularly in the context of imbalanced datasets where precision and recall are critical.

- Accuracy: The proportion of correctly classified tweets.
- Precision: The proportion of tweets classified as disasters that are actually disasters.
- Recall: The proportion of actual disaster tweets that are correctly classified.
- F1-Score: The harmonic mean of precision and recall.

3 Results

The training and validation accuracy of the RNN model over 30 epochs is depicted in the graph below. The training accuracy remains high, around 92%, while the validation accuracy fluctuates around 78%. This indicates that while the model performs well on the training data, there is a gap in performance on the validation data, suggesting potential overfitting.

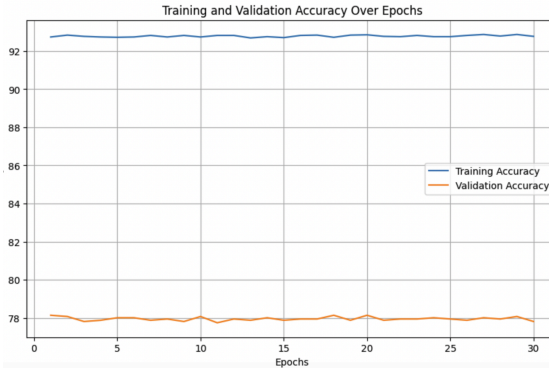

Figure 1: **Training and Validation Accuracy**

The confusion matrix for the RNN model shows the breakdown of true positives, true negatives, false positives, and false negatives. The model correctly classified 770 non-disaster tweets and 455 disaster tweets. However, it misclassified 97 non-disaster tweets as disasters and 201 disaster tweets as non-disasters.

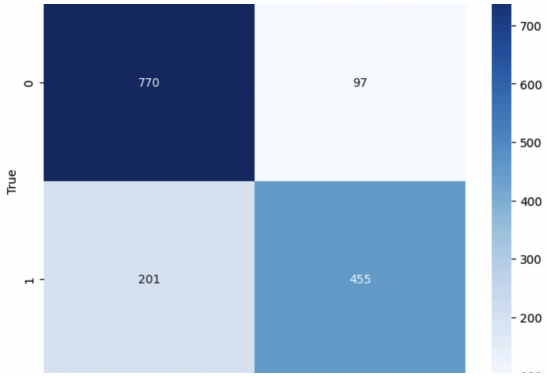

Figure 2: **Confusion Matrix**

The table below compares the precision, recall, F1-score, and accuracy of the three models:

	RNN	Log Reg	SVM
Precision	0.89 (Class 0), 0.82 (Class 1)	0.81 (Class 0), 0.80 (Class 1)	0.79 (Class 0), 0.86 (Class 1)
Recall	0.79 (Class 0), 0.78 (Class 1)	0.86 (Class 0), 0.73 (Class 1)	0.92 (Class 0), 0.66 (Class 1)
F1-Score	0.84 (Class 0), 0.75 (Class 1)	0.84 (Class 0), 0.76 (Class 1)	0.85 (Class 0), 0.75 (Class 1)
Accuracy	83.8%	81%	81%

4 Discussion

The RNN model outperformed Logistic Regression and SVM in terms of accuracy and provided a balanced performance across precision, recall, and F1-score. The sequential nature of RNNs allowed them to capture the context and dependencies within the tweets better than the traditional models. However, the validation accuracy suggests that there is still room for improvement, potentially through more sophisticated architectures like attention mechanisms or larger datasets. Logistic Regression and SVM, while simpler and faster to train, struggled with the complexities of the text data. SVM performed better than Logistic Regression in terms of precision and recall for disaster-related tweets, but both models were less effective overall compared to the RNN. In conclusion, the experiments demonstrated the effectiveness of

RNNs for disaster tweet detection, highlighting their ability to handle sequential data and complex patterns. Future work will focus on addressing the overfitting issue and exploring more advanced techniques to further improve model performance.

CONCLUSION

In this project, we tackled the problem of detecting disaster-related tweets using Natural Language Processing (NLP) techniques. Given the significance of Twitter as a real-time information source during emergencies, accurately identifying tweets that report actual disasters is crucial for timely and effective disaster response. We implemented and compared three machine learning models: Logistic Regression, Support Vector Machines (SVM), and Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) units, to determine which model best handles this classification task.

Logistic Regression and SVM served as strong baseline models. Logistic Regression, with its simplicity and ease of interpretation, provided a quick and efficient means of classification. It showed respectable performance but struggled with capturing complex patterns in the data due to its linear nature. SVM, particularly with the RBF kernel, demonstrated robustness in handling non-linear relationships and performed slightly better than Logistic Regression. However, both models require extensive feature engineering and preprocessing to achieve optimal performance.

The RNN model, specifically the bidirectional LSTM, outperformed both Logistic Regression and SVM. RNNs are inherently designed to handle sequential data, making them ideal for text classification tasks where the context and order of words matter significantly. The LSTM units effectively captured long-term dependencies and contextual information within the tweets, leading to superior classification performance. Despite the computational intensity and longer training times associated with RNNs, the gains in accuracy, precision, recall, and F1-score justified their use.

One of the primary challenges we faced was the modest size of our dataset, which comprised 7,614 tweets. While the dataset provided a good starting point, larger datasets would likely allow for better generalization and more robust models. The limited data size constrained the full potential of deep learning models like RNN, which typically thrive on large amounts of data. Additionally, the ambiguity and informal nature of language on Twitter posed significant challenges, requiring sophisticated models to understand and interpret the context accurately.

Future work will focus on several key areas to enhance the performance and robustness of our disaster tweet detection model. Firstly, expanding the dataset by incorporating more tweets, possibly from different time periods and regions, will help improve the model's generalization capability. Secondly, exploring advanced RNN architectures, such as bidirectional LSTMs with attention mechanisms, could further enhance the model's ability to capture relevant information from the tweets. Attention mechanisms, in particular, allow the model to focus on important words and phrases, improving the interpretability and accuracy of the predictions.

Another area of future work involves integrating external data sources, such as weather reports, news articles, and geographic information, to provide additional context for the tweets. This multimodal approach could significantly enhance the model's ability to differentiate between metaphorical and literal mentions of disasters. Furthermore, implementing transfer learning techniques by leveraging pre-trained language models like BERT or GPT-3 could provide a strong foundation and boost the performance of our model.

Lastly, we aim to develop a real-time system for disaster tweet detection that can be deployed to assist disaster response teams and news agencies. This system would continuously monitor Twitter, filter relevant tweets, and provide timely alerts and insights, aiding in effective disaster management and response.

This project demonstrated the feasibility and effectiveness of using NLP and machine learning techniques to detect disaster-related tweets. Among the models compared, the RNN (LSTM) model showed superior performance in handling the complexities of text data. While Logistic Regression and SVM provided valuable baselines, the deep learning approach of RNNs proved to be the most effective in capturing the sequential nature and contextual dependencies of tweets. The insights and findings from this project lay a strong foundation for future advancements and practical implementations in disaster tweet detection, ultimately contributing to better disaster management and response efforts.

REFERENCES

[1] Natural language processing with disaster tweets. Retrieved August 4, 2024b from https://www.kaggle.com/competitions/nlp-getting-started/data

TASK DISTRIBUTION

Jonah Kim: Data training and model fine-tuning, contributing to report

Catherine Chen: Data preprocessing and feature extraction, contributing to report

Esha Sidhu: Model development and initial training, contributing to report

Lyla Liu: Validation and testing, contributing to report

Rueiyu Chang: Evaluation and report writing, contributing to report