Natural Language Processing with Disaster Tweets

Alekya Rama & Namitha Ramakrishnan

Problem Statement

Objective: To Classify tweets as disaster-related (1) or not (0)

Goal:

- Use AI & NLP to distinguish real crisis events from casual tweets.
- Helps emergency teams filter urgent information faster.

Disaster - Related Tweet

Non Disaster- Related Tweet

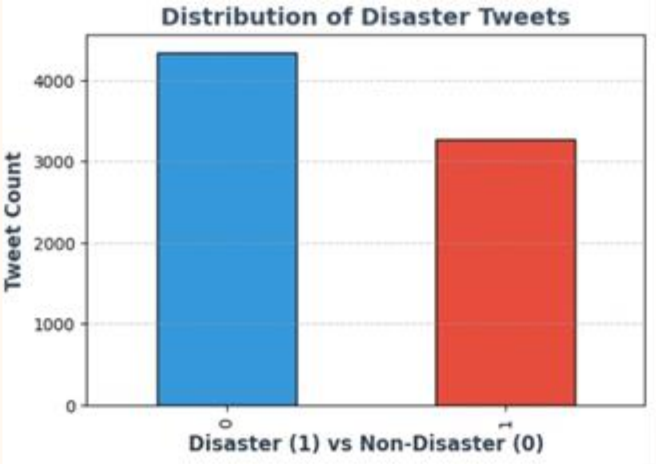
Dataset Overview

Source: Kaggle's NLP Disaster Tweets dataset

Train Data: **7,613** labelled tweets

- Disaster (1): 43%
- Non -Disaster (0: 57%)

Test Data: **3,263** unlabelled tweets (for evaluation)



Key Data Challenges

Noisy Text:
Misspellings, slang, emojis, and hashtags

Short Texts:
280-character limit reduces context

Figurative Language:
Hard to distinguish real disasters from expressions

Missing Data:
Some tweets lack location or keyword

Data Pre Processing

Why Preprocessing Matters?

Raw tweets contain noise such as:

Special characters & punctuation	"Help!!! Flooding in Texas 😢 😢 "
URLs & mentions	"Check this out! https://xyz.com @user"
Stopwords	"is", "the", "at", "on" (common but non-informative words)
Inconsistent casing	"FIRE" vs. "fire"

Data Pre Processing

Text Cleaning	Tokenization	Stopword Removal	Lemmatization/ Stemming	Vectorization	Padding & Truncation
✓ Convert to lowercase → standardizes text ✓ Remove punctuation, URLs, hashtags & mentions	✓ Breaks tweet into words e.g: "fire in city" → ["fire", "in", "city"]	✓ Removes common but unimportant words e.g., "a", "the", "is"	✓ Converts words to their base form e.g.: "running" → "run"	✓ Converts words to numbers ✓ TF-IDF / Count Vectorization For traditional ML models ✓ Word Embeddings (GloVe, Word2Vec) For Deep Learning	✓ Tweets have different lengths , standardize to fixed length ✓ Use max_length=50 , Ensures consistent input for models

Model Performance Evolution

Model No.	Model Type	Accuracy	Key Enhancements
1	Baseline CNN/LSTM	0.770	✅ Standard Text Pre processing ✅ Simple architecture
2	CNN + LSTM	0.773	✅ Added dropout ✅ Batch normalization ✅ Improved embeddings
3	Ensembler Model – CNN + LSTM	0.81	✅ Increased LSTM Units, ✅ Increased Regularization ✅ Improved Learning Rate

Model 1 – BiDirectional LSTM Model (Accuracy: 0.770)

Features:

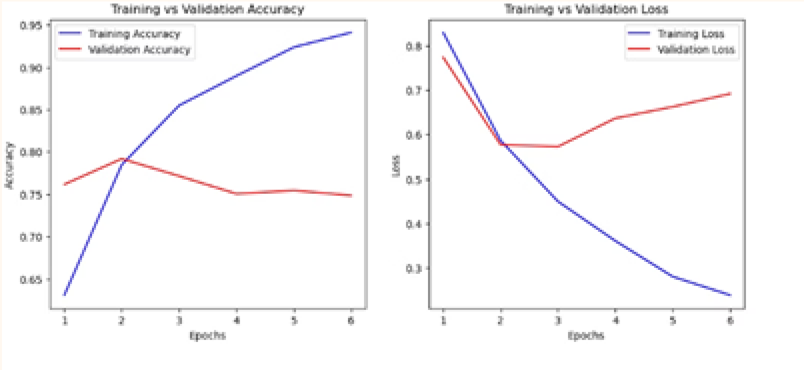
- Forward and Backward Processing
- Embedding Layer
- Dropout & Recurrent Dropout
- Stacked LSTM Layers
- Dense Output Layer (Sigmoid Activation)

Training & Validation Analysis:

- Good training accuracy, but validation accuracy plateaued
- Overfitting observed, as validation loss increased after a few epochs

Key Takeaways

- More dropout, L2 regularization, or even batch normalization layers can help reduce overfitting.
- If overfitting persists, consider reducing the model's capacity (e.g., fewer neurons or layers) or further tuning hyperparameters.
- Always monitor validation accuracy and loss. Relying on training metrics alone can be misleading.



Model 2 – Improved CNN + LSTM (Accuracy: 0.773)

Features:

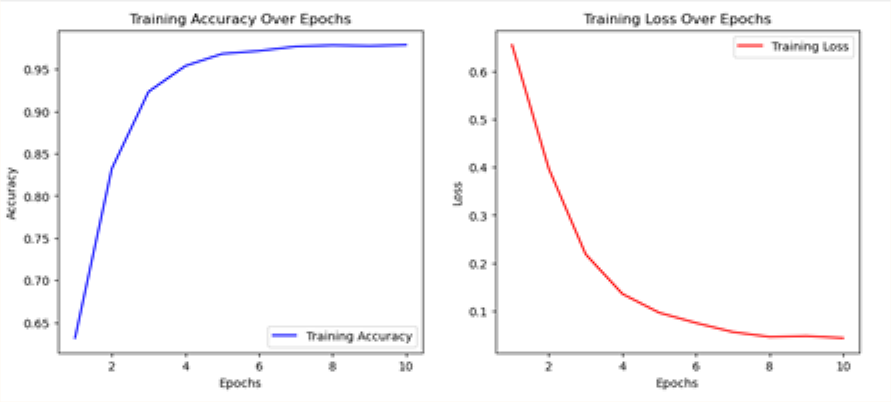
- Convolutional layers extract local patterns
- LSTM layer captures long-range dependencies in text
- Dropout (0.5) & L2 Regularization to control overfitting
- Optimized learning rate and batch size

Training & Validation Analysis:

- Slightly better generalization than previous model
- Reduced overfitting, but validation accuracy still fluctuated

Key Takeaways

- Better sequential understanding using LSTM.
- Still some overfitting—could be improved with better regularization



Model 3 – CNN + BiLSTM (Accuracy: 0.810)

Features:

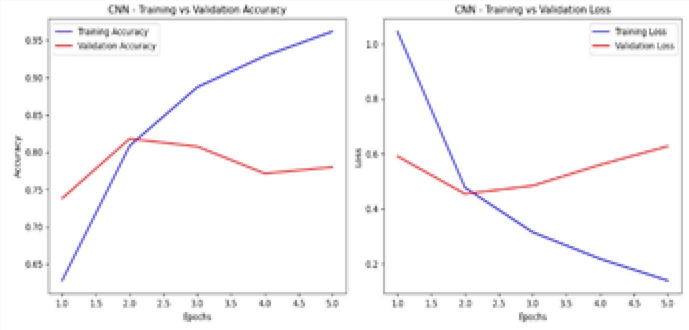
- Combined predictions from CNN & LSTM models
- Averaged final outputs instead of selecting a single model
- Early stopping & adaptive learning rate for optimization

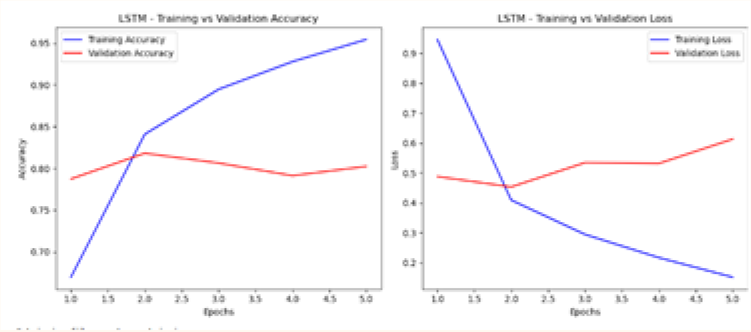
Training & Validation Analysis:

- More stable validation accuracy, less overfitting
- Smoother loss curve compared to others
- Stronger robustness to tweet variations

Key Takeaways

- Best-performing model so far—ensemble approach improved generalization
- Still not 100% accuracy—potential for BERT or Transformer-based models



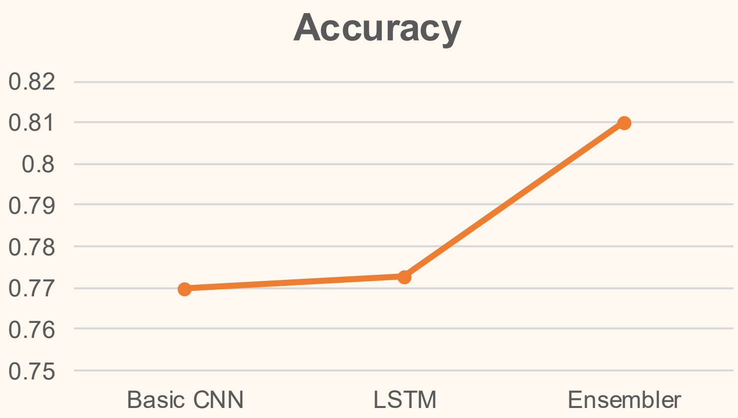


Key Takeaways:

Model Evolution:

CNN → LSTM → ENSEMBLER

Each step improved accuracy & robustness of the model



- **Overfitting Reduction -** Dropout layers, regularization techniques helped.
- **Generalization -** The ensemble model performed better on unseen data.