NLP Pipeline for Disaster Response Message Classification and Analysis

Leo Davidov
leo.davidov@student.oulu.fi
2504435

Bharathi Sekar
bharathi.sekar@student.oulu.fi
2409295

Chamudi Vidanagama
chamudi.vidanagama@student.oulu.fi
2405189

TA in charge: Fuzel Shaik
The Center for Machine Vision and Signal Analysis
University of Oulu, Finland
fuzel.shaik@oulu.fi

Abstract

This project presents a comprehensive natural language processing pipeline for multi-label classification and analysis of disaster response messages, using the publicly available Disaster Response Messages dataset. A range of traditional and deep learning models were evaluated to assess their effectiveness in handling short, noisy, and imbalanced textual data. Classical linear models such as Logistic Regression and LinearSVC trained on TF-IDF and BoW representations demonstrated a good baseline performance, achieving macro-F1 scores around 0.40–0.45 in multi-label classification of 35 targets. Deep neural networks, including dense multilayer perceptrons and bidirectional LSTM architectures were also examined along with BERT transformer. Beyond classification, auxiliary analyses such as Named Entity Recognition (NER), Topic Modeling with Latent Dirichlet Allocation (LDA) and BERTopic, along with Sentiment Analysis using VADER, and TextBlob provided interpretable insights into location, emotional tone, and emerging disaster themes. The integrated approach demonstrates that traditional vector-based models remain competitive for low-resource crisis text.

Code is publicly available at: https://github.com/littlemicrowave/NLP-disaster-messages

Keywords: NLP, Topic Modeling, Deep Learning, Machine Learning, NER, LDA, Sentiment analysis, Text Preprocessing, Message Classification

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

1 Related Work

Disaster response message classification is an established yet evolving domain intersecting Natural Language Processing (NLP) and humanitarian response analytics. Prior studies such as those by Imran et al. [1] and Nguyen et al. [2] explored the automated categorization of social media crisis data using logistic regression and SVM-based frameworks, demonstrating the early feasibility of machine learning in emergency communication filtering.

Traditional approaches rely heavily on bag-of-words (BoW) and term frequency–inverse document frequency (TF-IDF) representations [3], which capture lexical frequency features but lack semantic awareness. The introduction of distributed word representations such as Word2Vec [4] and GloVe [5] enabled models to encode semantic similarity in continuous vector spaces, improving downstream classification accuracy and generalization.

Subsequent deep learning architectures, particularly recurrent networks such as LSTM [6] and transformer-based models including BERT [7] and its compact variant DistilBERT [8], have further advanced text understanding through contextual embeddings. These models have been shown to outperform classical baselines in various crisis-related NLP benchmarks by leveraging bidirectional context and pretraining on large-scale corpora.

Beyond classification, several complementary NLP techniques have been successfully applied to crisis informatics. Topic modeling methods, including probabilistic models like Latent Dirichlet Allocation (LDA) [9] and modern embedding-based approaches such as BERTopic [10], have been used to extract latent themes and event clusters from large volumes of social media data during disasters. Similarly, Named Entity Recognition (NER) has proven effective for extracting location, organization, and person entities from unstructured text, enabling geographic triage and coordination support. Sentiment analysis tools such as VADER [11] and TextBlob [12] have also been incorporated to quantify emotional tone and urgency within emergency communications.

2 Methodology

2.1 Overall Pipeline Architecture

The project implemented a comprehensive end-to-end Natural Language Processing pipeline designed specifically for disaster response message analysis. It is structured around ten distinct tasks that progressed systematically from data acquisition through advanced analysis. The pipeline followed a systematic workflow encompassing data acquisition, preprocessing, feature extraction, model training, evaluation, and multi-faceted insight extraction. The architecture was structured to handle the unique challenges of disaster messages, including short text length, multi-label classification requirements, class imbalance, and the need for interpretable results for emergency responders.

2.2 Dataset Characteristics and Initial Processing

The dataset comprised 26,216 messages collected from real disaster events including earthquakes, floods, and hurricanes, provided through the Kaggle platform "Disaster Response Messages" [13]. The data was structured across two primary files which are messages containing the raw text and metadata, and categories containing the multi-label annotations across 36 disaster response categories. The dataset showed significant class imbalance, with the `related` category containing 20,094 instances while rare categories like `offer` had only 118 instances.

Initial data quality assessment identified 68 duplicate messages and 32 duplicate category labelings that were removed to prevent data leakage. The `original` column contained 16,064 missing values, which were determined to represent messages already in English that didn't require translation. Data quality issue was discovered in the `related` column, where 193 instances contained the value "2" instead of the expected binary values. After investigation, these were normalized to value "1" to maintain consistency. The `child_alone` category was removed entirely as it contained only one unique value, providing no discriminative power for classification.

2.3 Text Cleaning and Preprocessing

We implemented a text cleaning strategy optimized for traditional machine learning models, recognizing that disaster messages often contain noise from various sources including social media formatting, translation artifacts, and informal communication styles. The preprocessing pipeline employed multiple stages of text normalization and filtering. All text was converted to lowercase to ensure consistency in token matching. A comprehensive regular expression pattern was developed to remove URLs, user mentions, hashtags, numbers, punctuation, and extra whitespace characters that could introduce noise without semantic value for classification.

Regex: `(https?://\S+|https?\s\S*|www\.\S+|\bhttps?\b)|[@#]\w+|\d+|[^\w\s]|_`

The preprocessing utilized the Natural Language Toolkit (NLTK) [14] for linguistic processing. Text was tokenized using the `word_tokenize` function to handle contractions and punctuation. Stop words were removed using an English stopword list to eliminate common words. Lemmatization was performed using `WordNetLemmatizer` [15] to reduce words to their base forms while preserving meaningful semantic content. The cleaned tokens were stored as lists for flexible downstream processing, with the option to reconstruct sentence representations when needed. This preprocessing approach was specifically designed to handle the dense nature of disaster messages while reducing vocabulary size and computational complexity for subsequent modeling stages.

2.4 Feature Extraction Methods

2.4.1 Text Representation Using Classical Methods

We implemented Bag-of-Words (BoW) and TF-IDF representations using `scikit-learn`'s `CountVectorizer` and `TfidfVectorizer` [16]. Both methods employed vocabulary filtering with `min_df=5` and `max_df=0.95` to eliminate rare (hapax) and overly common terms, resulting in a refined vocabulary of 6,224 features. The BoW approach captured term frequencies as integer counts, while TF-IDF applied logarithmic scaling to emphasize discriminative terms through inverse document frequency weighting.

The TF-IDF score for a term t in a document d is given by:

$$\text{TF-IDF}(t, d) = \text{tf}(t, d) \times \log \frac{N}{\text{df}(t)}$$

where $\text{tf}(t, d)$ is the frequency of term t in document d, $\text{df}(t)$ is the number of documents containing t, and N is the total number of documents.

We used unigram models `ngram_range=(1,1)` since preliminary analysis showed limited benefit from higher-order n-grams, and inflated vocabulary size, which impacted the computational time. All vectorizers were fit on training data (80%–20% train-test split) to prevent data leakage.

2.4.2 Word Embedding Representations

We trained a custom Word2Vec model using Gensim [4] with skip-gram architecture, generating 50-dimensional word embeddings with a window size of 5 over 10 training epochs. The model achieved vocabulary coverage of 6,366 terms. For traditional ML compatibility, we created sense vectors by summing word embeddings per document followed by normalization of the result, producing fixed 50-dimensional sense-carrying vector representations of messages:

$$\vec{d} = \frac{\sum_{w \in W_d} \vec{w}}{\left\| \sum_{w \in W_d} \vec{w} \right\|_2}$$

For sequence models, we implemented truncation to create fixed-length sequences of 50 word embeddings, preserving word order information while maintaining consistent input dimensions across all messages. This dual approach enabled both efficient traditional modeling and sophisticated sequential analysis with LSTM and BERT.

2.5 Classical Machine Learning Classification

Further in our pipeline we have applied traditional supervised machine-learning algorithms to perform multi-label classification of messages into 35 target classes using the obtained TF-IDF, BOW vector representations of messages, and dense message embeddings (sense vectors) derived from Word2Vec embeddings as stated in paragraph above. For the training and evaluation of models, the machine learning library scikit-learn [16] was used, which provides all necessary tools. Given that each message may belong to multiple categories, the problem formulation corresponds to multi-label classification, where the overall model comprises N independent binary classifiers, one for each label.

Performance of classical classifiers was examined on each feature set to model the multi-label text classification problem:

1. **Linear Support Vector Machine (SVM):** Implemented using `LinearSVC` wrapped in a `MultiOutputClassifier` to enable independent binary decision functions per label. A grid search over the regularization parameter $C \in \{0.1, 1, 10\}$ with L2 penalty was conducted using a `PredefinedSplit` [17].

2. **Logistic Regression (LR):** Also applied via `MultiOutputClassifier`, optimized with the `lbfgs` solver and L2 regularization. A hyperparameter search for $C \in \{0.01, 0.1, 1, 10\}$ was executed to find the best value which ensures generalization.

3. **Random Forest (RF):** A non-linear ensemble baseline was included to compare the performance of tree-based models against linear ones. Hyperparameters such as number of estimators ($n\_estimators \in \{200, 300\}$), minimal split size ($min\_samples\_split \in \{2, 5, 10\}$), and leaf size ($min\_samples\_leaf \in \{2, 5\}$) were tuned using grid search with class balancing enabled, to improve generalization and prevent uncontrollable tree growth during the training [18].

2.6 Deep Learning Based Classification

In addition to classical machine learning models, deep learning techniques were explored to examine potential architectural benefits for better semantic and contextual relationship modeling in text messages. Four architectures were implemented and evaluated: (1) A simple dense MLP for TF-IDF and sense vector representations, (2) a bidirectional Long Short-Term Memory (LSTM) recurrent neural network [6] for word vector sequences from previously pre-trained Word2Vec model, (3) a bidirectional LSTM model with a trainable embedding layer, and (4) a fine-tuned distilled BERT transformer model [8]. All models were trained for multi-label text classification, where each message could belong to multiple categories, and using binary cross-entropy loss with evaluation metrics of macro-, micro-, and weighted-F1 scores to account for class imbalance. We have additionally examined LSTM and distilled BERT transformer performance on semi-raw message strings, without stop-word removal and lemmatization applied.

2.6.1 Dense Model

This baseline deep model extends the classical approach by replacing linear classifiers with a non-linear dense neural network. TF-IDF vectors served as inputs to a multi-layer perceptron (MLP) consisting of two fully connected layers (256 and 128 neurons) with ReLU activation and a dropout rate of 0.2. The output layer used 35 sigmoid-activated neurons to enable multi-label prediction. The model was trained using the Adam optimizer [19] with learning rates between 0.001 and 0.0001, batch sizes of 32–64, and for up to 10 epochs. This architecture primarily served as a baseline to assess whether non-linear transformations over TF-IDF features could improve upon classical SVM and Logistic Regression performance. In addition, we have tested the sense vector representations on this architecture. The model was implemented in TensorFlow. Model graph is presented below.

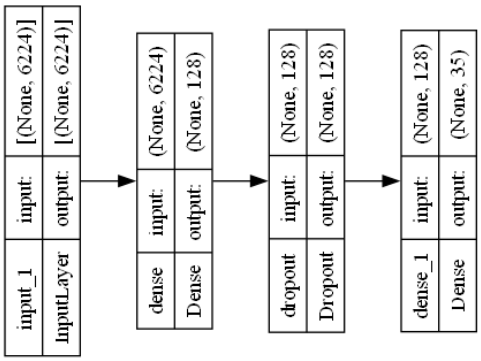

Figure 1: Dense MLP model

2.6.2 biLSTM Model

To capture semantic meaning and word-level context, a BiLSTM model was implemented using 50-dimensional pre-trained Word2Vec embeddings. The model architecture included a bidirectional LSTM layer with 128 hidden units in each direction, followed by a dense layer of 128 neurons and a dropout of 0.3. The output layer again consisted of 35 sigmoid-activated neurons. The model was implemented in TensorFlow. Model graph is presented below.

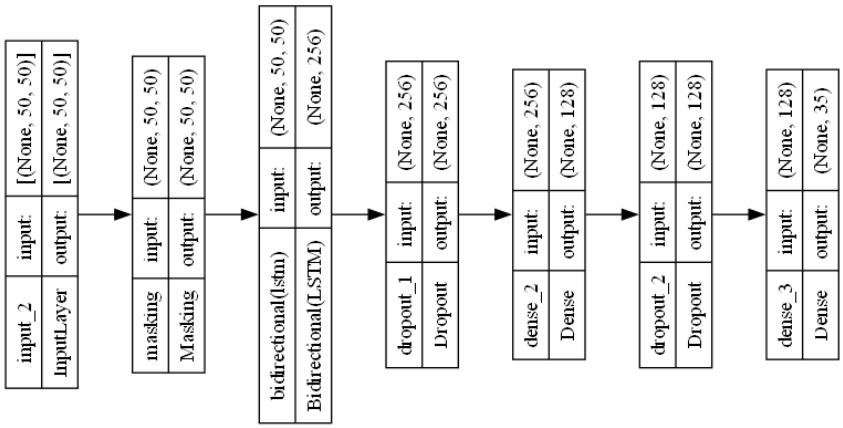

Figure 2: Bidirectional LSTM model

2.6.3 Trainable Embeddings + biLSTM Model

A second BiLSTM variant was developed with a randomly initialized embedding layer that was fully trainable during model optimization. The embedding size was set to 50, with the same sequence length of 50 tokens, and the vocabulary size for the embedding matrix was set to 8000 with BiLSTM dimensions of 128 units per direction. This model allows the embeddings to adapt to the task-specific vocabulary. The model was implemented in TensorFlow. Graph is presented below.

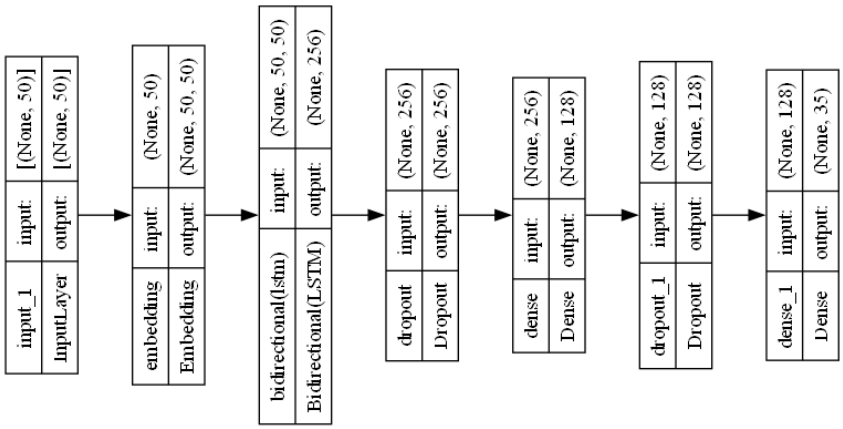

Figure 3: Bidirectional LSTM model with trainable embedding layer

2.6.4 Distilled BERT Transformer

To improve the model's understanding of semantic nuances and contextual relationships, a DistilBERT transformer model was fine-tuned for multi-label classification. DistilBERT is a 66-million parameter version derived from BERT through knowledge distillation, which retains much of BERT's linguistic understanding while being more computationally efficient and faster to train [8]. Each message was tokenized using the DistilBERT tokenizer with a maximum sequence length set to 50 using truncation. Instead of the `[CLS]` token for classification, we have employed mean pooling, the average of all token embeddings in the final hidden layer, as it was giving more stable results. This pooled embedding was passed through a dropout layer (rate 0.3), a dense layer of 256 neurons with ReLU activation, and finally a 35-neuron sigmoid output layer. Training was performed with the AdamW optimizer with learning rate 2×10^{-5}, batch size 64, and 10 epochs. The model was trained and customized in PyTorch.

2.7 Named Entity Recognition

For Named Entity Recognition, we employed spaCy's pre-trained `en_core_web_sm` model [20]. The pipeline processed all 26,216 cleaned texts, with entity extraction performed in batches of 64 using multiprocessing to leverage available CPU cores. We extracted standard entity types including `GPE` (Geopolitical Entities), `DATE`, `TIME`, `ORG` (Organizations), `PERSON`, and various numerical entities. The NER system was configured to process the cleaned text versions, recognizing that our preprocessing might affect entity recognition performance by removing capitalization and punctuation cues that spaCy's model typically leverages.

We implemented comprehensive entity frequency analysis across the entire corpus and developed sampling strategies to identify messages with rich entity content for qualitative analysis. The system also supported entity visualization using spaCy's `displaCy` module for manual inspection and validation of extraction quality.

2.8 Topic Modeling

We implemented two distinct topic modeling approaches: traditional Latent Dirichlet Allocation (LDA) and modern BERTopic. For LDA, we used the `CountVectorizer` with the same parameters as the classification pipeline (`min_df=5`, `max_df=0.95`) to create the document-term matrix. LDA was trained with 10 topics using the scikit-learn's implementation `LatentDirichletAllocation` with `random_state=42` for reproducibility and `max_iter=10` for computational efficiency [9].

BERTopic employed a more sophisticated pipeline using sentence transformers for document embeddings, UMAP for dimensionality reduction, and KMeans for clustering [10]. We configured UMAP with `n_neighbors=15`, `n_components=5`, and `min_dist=0.001` to preserve local structure while reducing dimensionality. Instead of HDBSCAN, we used KMeans with 10 clusters for more con-

trolled topic count. The `top_n_words` parameter was set to 15 for detailed topic interpretation, and the model was configured to automatically reduce topics to the specified count.

2.9 Sentiment Analysis

For sentiment analysis, we implemented two complementary approaches: VADER (Valence Aware Dictionary and Sentiment Reasoner) and TextBlob. VADER is specifically optimized for social media text and short messages [11]. We applied VADER's `polarity_scores` method to extract compound sentiment scores, with threshold-based labeling (positive: ≥ 0.05, negative: ≤ -0.05, neutral: between).

TextBlob provided an alternative lexicon-based approach using its `pattern` library sentiment analysis [12]. We extracted polarity scores ranging from most negative to most positive, with binary thresholding at zero for label assignment. Both systems processed the cleaned text versions, allowing direct comparison of their sensitivity to the same input representations.

We implemented comprehensive aggregation of sentiment scores by disaster category, enabling analysis of emotional patterns across different types of disaster communications and response needs.

3 Results

3.1 Data Characteristics and Class Distribution

The final dataset contained 26,216 messages across 35 categories after data cleaning and preprocessing. A severe class imbalance was observed, with the `related` category dominating, followed by `aid_related` and `weather_related`. In contrast, critical but rare categories such as `search_and_rescue`, `fire`, and `missing_people` contained only a few hundred instances each.

A histogram of categories per message revealed the inherently multi-label nature of the problem. Reflecting the complex, overlapping semantics of real-world disaster communication. 20% of dataset messages were not assigned to any category, those are messages which are marked as **not** `related`, and about 20% more of dataset is assigned only to disaster `related` but miss any other specifying category label, also five top category labels: `related`, `aid_related`, `weather_related`, `direct_report`, `request`, `other_aid` correspond to about half of category markings. About 40% of messages is marked to `aid_related` and about 25% is `weather_related`, so we definitely see that there are two dominant specifiying categories.

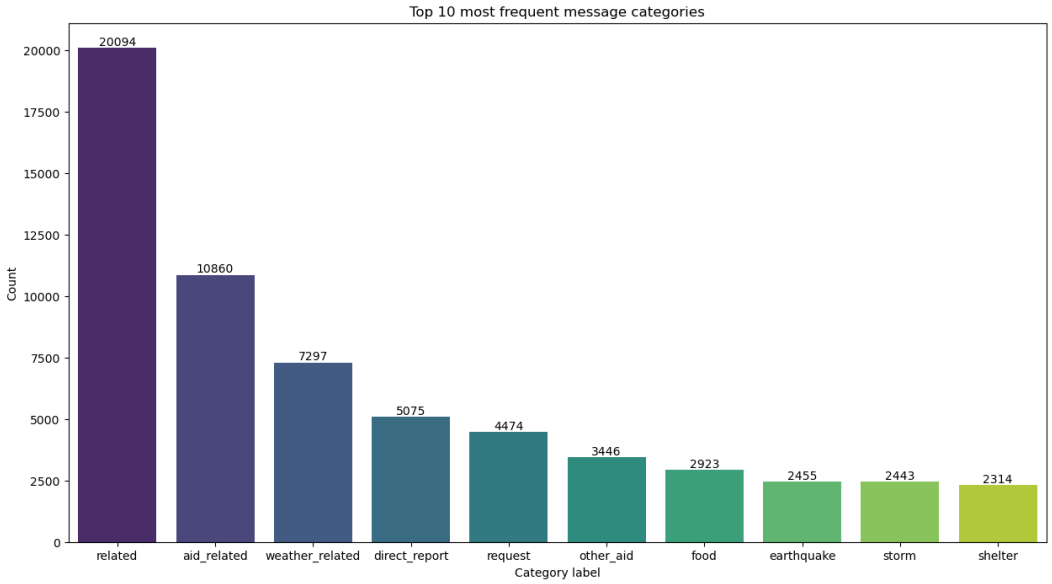

Figure 4: Bar chart of top 10 category frequencies

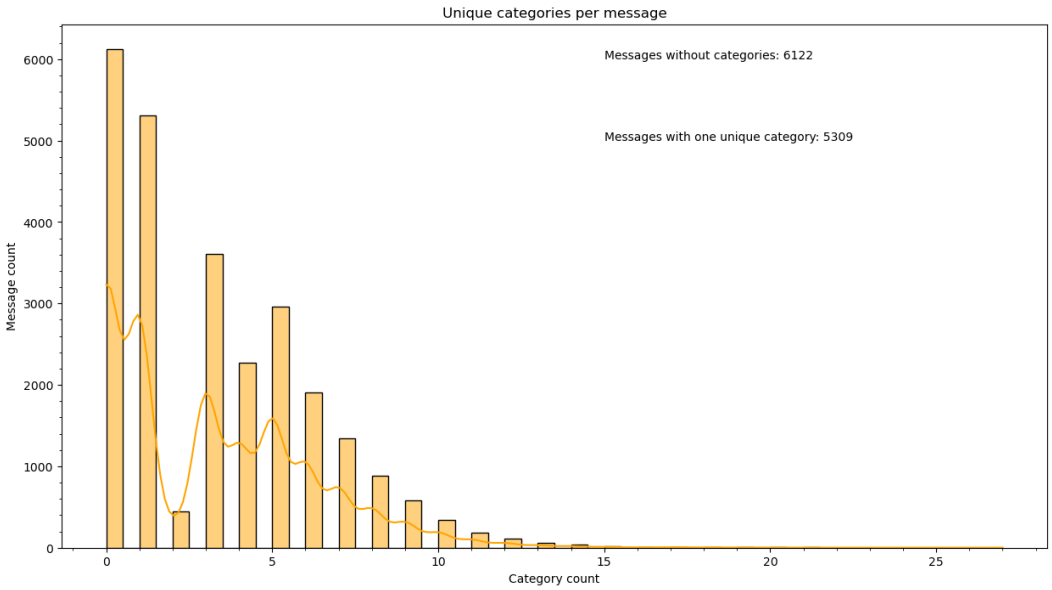

Figure 5: Histogram of categories per message

This imbalance and sparsity motivated the use of evaluation metrics that are robust to class imbalance, such as the macro- and weighted-F1 scores, and justified additional analysis of the `related` class as a potential confounder in downstream classification tasks.

3.2 Text Cleaning and Preprocessing Results

The text preprocessing pipeline transformed raw, noisy messages into normalized token sequences suitable for both classical and deep learning models. Each message was processed through the following transformations:

1. Conversion to lowercase: standardizing capitalization for consistent tokenization.
2. Regular-expression based removal of URLs, user mentions, numbers, and punctuation.
3. Tokenization using a rule-based tokenizer.
4. Stopword removal to eliminate non-informative words.
5. Lemmatization to reduce inflectional forms to canonical lemmas.

Sample transformations demonstrated effective noise removal while preserving semantic content. For example:

> `Raw:` *UN reports Leogane 80-90% destroyed!*
> `Cleaned:` [“un”, “report”, “leogane”, “destroyed”]

The processed tokens were stored in a new `clean_text` column, ensuring compatibility across downstream tasks.

Table 1: Examples of message preprocessing

Raw Message	**Cleaned Tokens**
“UN reports Leogane 80-90 destroyed.”	`[un, report, leogane, destroyed]`
“Need food and water ASAP!”	`[need, food, water]`
“There’s a fire near hospital area.”	`[fire, near, hospital, area]`

These preprocessing results demonstrate substantial noise reduction, vocabulary normalization, and improved interpretability of token-level inputs for subsequent feature extraction and modeling.

3.3 Task 3: Feature Space Analysis

Bag-of-Words and TF-IDF vectorizers produced 6,224 features after filtering (`min_df=5`, `max_df=0.95`), creating sparse matrices of dimensions 20,972 $\times$ 6,224 (20,972 is amount of documents in train set). Frequency analysis identified top terms: `water` (3,039), `people` (3,013), `food` (2,902), `help` (2,650), `need` (2,491). Word2Vec trained 50-dimensional embeddings for 6,366 terms using skip-gram architecture. Message length analysis showed 99% of messages contained fewer than 50 tokens (median: 12 tokens). PCA visualization of embeddings revealed semantic clustering of related disaster terms.

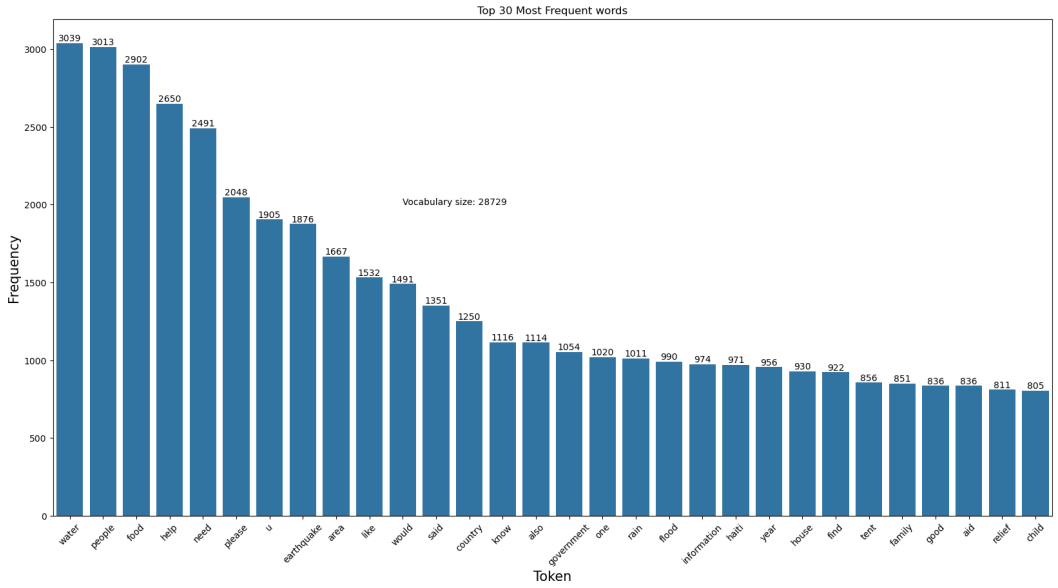

Figure 6: Top 30 words frequency bar plot

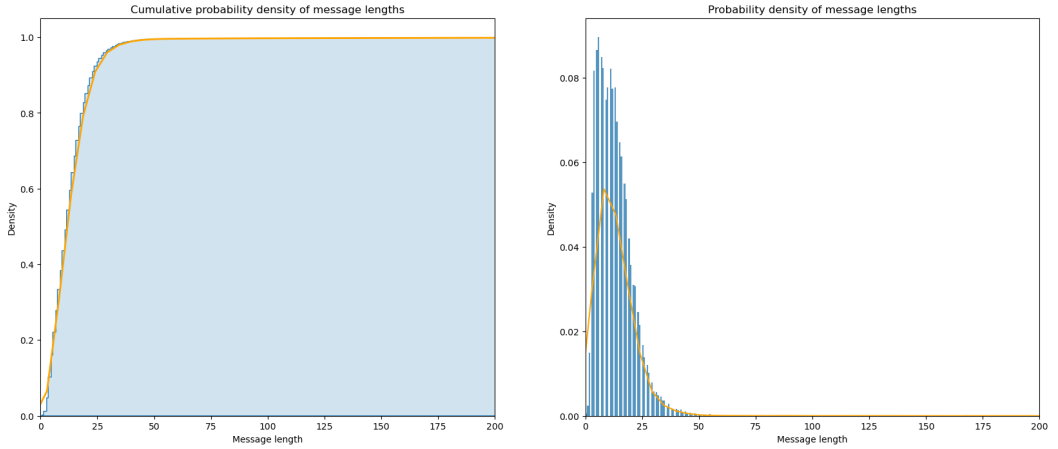

Figure 7: Message length distribution

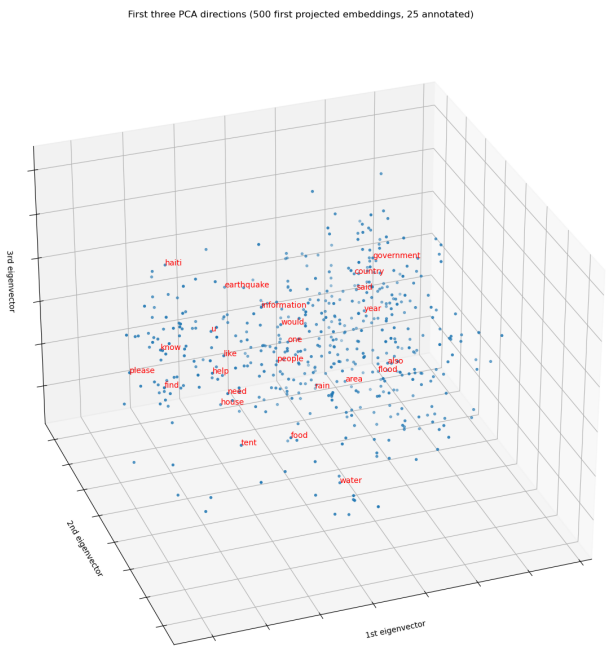

Figure 8: PCA projection of Word2Vec embeddings

3.4 Machine Learning Classification

Among the tested models, linear classifiers (SVM and Logistic Regression) demonstrated the strongest performance and generalization capability across feature representations. Both achieved macro-F1 scores in the range of 0.40–0.45 and weighted-F1 around 0.60–0.65, indicating consistent performance across both frequent and infrequent classes. In contrast, the Random Forest classifier underperformed significantly, with macro-F1 values between 0.10–0.20, reflecting its limited ability to learn from high-dimensional, sparse text features.

BoW features slightly outperformed TF-IDF counts across linear models, typically improving F1 scores by 0.01–0.02. Conversely, the sense vector embeddings (50-dimensional dense representations) led to a noticeable drop in performance across all models in classification of rare labels, while getting a slight gain 0.01–0.02 in common ones, likely due to excessive compression of message semantics and loss of category-specific lexical cues. A brief summary of classification results is presented on the heatmap of F1-scoring types across features and models.

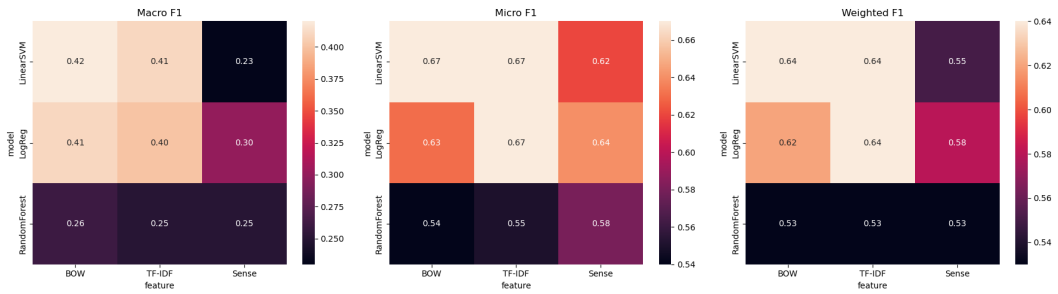

Figure 9: Heatmap of general classification results

3.4.1 Support Vector Machine (LinearSVC)

Table 2: SVM BOW classification results

Category/Metric	Precision	Recall	F1-Score	Support
related	0.86	0.90	0.88	4018
micro avg	0.77	0.60	0.67	16821
macro avg	0.59	0.35	0.42	16821
weighted avg	0.73	0.60	0.64	16821
samples avg	0.61	0.50	0.50	16821

Table 3: SVM TF-IDF classification results

Category/Metric	Precision	Recall	F1-Score	Support
related	0.87	0.90	0.88	4018
micro avg	0.77	0.59	0.67	16821
macro avg	0.62	0.34	0.41	16821
weighted avg	0.73	0.59	0.64	16821
samples avg	0.60	0.50	0.50	16821

Table 4: SVM Dense sense vector classification results

Category/Metric	Precision	Recall	F1-Score	Support
related	0.84	0.95	0.89	4018
micro avg	0.78	0.52	0.62	16821
macro avg	0.44	0.19	0.23	16821
weighted avg	0.68	0.52	0.55	16821
samples avg	0.64	0.48	0.50	16821

The LinearSVC classifier was optimized over $C \in \{0.1, 1, 10\}$ using L2 regularization. The best parameter combination used $C = 0.1$ with BoW features. BoW + SVM achieved a macro-F1 of 0.42 and weighted-F1 of 0.64, slightly outperforming the TF-IDF in F1-scoring, and significantly sense-vector variants. The model showed high recall for frequent categories such as `related` (0.90), `aid_related` (0.66), and `weather_related` (0.71), but struggled with rare labels (`offer`, `shops`, `hospitals`). 35 confusion matrices (figures omitted due to size) for binary labels showed strong separation for dominant categories, with most misclassifications occurring among semantically similar classes (`medical_help`, `medical_products`, `other_aid`). Overall, the linear SVM effectively leveraged sparse BoW and TF-IDF features.

3.4.2 Logistic Regression

Table 5: Logistic Regression BoW classification results

Category/Metric	Precision	Recall	F1-Score	Support
related	0.87	0.87	0.87	4018
micro avg	0.67	0.60	0.63	16821
macro avg	0.49	0.37	0.41	16821
weighted avg	0.65	0.60	0.62	16821
samples avg	0.54	0.49	0.47	16821

Table 6: Logistic Regression TF-IDF classification results

Category/Metric	Precision	Recall	F1-Score	Support
related	0.87	0.91	0.89	4018
micro avg	0.77	0.59	0.67	16821
macro avg	0.62	0.33	0.40	16821
weighted avg	0.73	0.59	0.64	16821
samples avg	0.61	0.50	0.50	16821

Table 7: Logistic Regression dense sense-vector classification results

Category/Metric	Precision	Recall	F1-Score	Support
related	0.84	0.94	0.89	4018
micro avg	0.77	0.55	0.64	16821
macro avg	0.52	0.24	0.30	16821
weighted avg	0.71	0.55	0.58	16821
samples avg	0.63	0.49	0.50	16821

The Logistic Regression model was trained with L2 regularization using the `lbfgs` solver and hyperparameter $C \in \{0.01, 0.1, 1, 10\}$. The BoW representation again performed best, achieving a macro-F1 of 0.41 and a weighted-F1 of 0.62 with $C = 10$. TF-IDF resulted in a minor decrease in macro-F1 (0.40) while maintaining comparable weighted averages. The dense sense-vector representation yielded reasonable performance for dominant categories but substantially lower macro-F1 (0.30), indicating weaker generalization across rare labels.

Despite its simplicity, the Logistic Regression model demonstrated stable results, effectively capturing linear relationships within the high-dimensional sparse feature space. It also maintained a balanced precision–recall trade-off for moderately frequent categories such as `request`, `food`, `shelter`, and `direct_report`.

3.4.3 Random Forest

Table 8: Random Forest BoW classification results

Category/Metric	Precision	Recall	F1-Score	Support
related	0.77	1.00	0.87	4018
micro avg	0.45	0.67	0.54	16821
macro avg	0.35	0.30	0.26	16821
weighted avg	0.51	0.67	0.53	16821
samples avg	0.42	0.59	0.44	16821

Table 9: Random Forest TF-IDF classification results

Category/Metric	Precision	Recall	F1-Score	Support
related	0.77	1.00	0.87	4018
micro avg	0.47	0.66	0.55	16821
macro avg	0.35	0.28	0.25	16821
weighted avg	0.52	0.66	0.53	16821
samples avg	0.44	0.58	0.45	16821

Table 10: Random Forest dense sense-vector classification results

Category/Metric	Precision	Recall	F1-Score	Support
related	0.77	1.00	0.87	4018
micro avg	0.53	0.64	0.58	16821
macro avg	0.40	0.26	0.25	16821
weighted avg	0.54	0.64	0.53	16821
samples avg	0.50	0.58	0.48	16821

The Random Forest classifier was optimized with $n\_estimators \in \{200, 300\}$, minimal split size min_samples_split $\in \{2, 5, 10\}$, and leaf size min_samples_leaf $\in \{2, 5\}$, with class balancing enabled. Even with balancing, performance remained notably weaker than for linear models. The BoW + RF combination achieved a macro-F1 of approximately 0.26 and a weighted-F1 of 0.53, while TF-IDF and dense sense-vector inputs yielded similar or slightly lower results.

Random Forest tended to overfit the training data, producing inflated recall but poor precision for dominant categories and failing to generalize to minority classes. These findings confirm that tree-based ensembles are not well suited to highly sparse, high-dimensional textual feature spaces typical of disaster-response message data.

4 Deep Learning classification

4.1 Dense Model on TF-IDF Features

Table 11: Dense MLP on TF-IDF

Category/Metric	Precision	Recall	F1-Score	Support
related	0.86	0.92	0.89	4018
micro avg	0.78	0.59	0.67	16821
macro avg	0.70	0.31	0.38	16821
weighted avg	0.75	0.59	0.63	16821
samples avg	0.75	0.66	0.58	16821

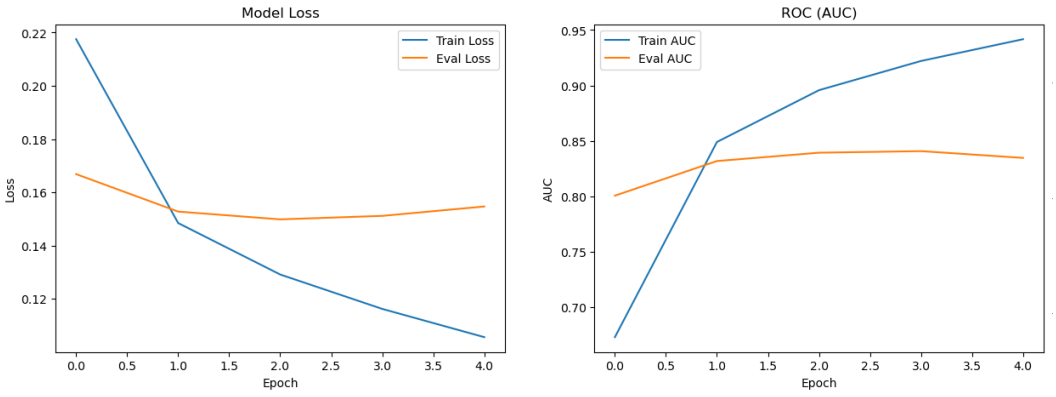

Figure 10: Dense MLP on TF-IDF training/evaluation graph

After 5 training epochs, the model reached a validation AUC of 0.84, after the second epoch model was starting to overfit training data. Result analysis showed strong performance on dominant labels such as `related` with F1 of 0.89, `aid_related` of 0.73, `food` 0.76, and `weather_related` 0.75. Macro-F1 and weighted-F1 scores were approximately 0.38 and 0.63, respectively, nearly matching the classical SVM baseline. Smaller or infrequent categories (e.g., `hospitals`, `tools`, `shops`)

remained challenging, with precision and recall near zero due to limited positive samples. These results suggest that TF-IDF features capture sufficient lexical signal for linear and shallow nonlinear models, effectively modeling category-specific word patterns.

4.2 Dense Model on Sense Embeddings

Table 12: Dense model on sense embeddings

Category/Metric	Precision	Recall	F1-Score	Support
related	0.84	0.95	0.89	4018
micro avg	0.77	0.57	0.65	16821
macro avg	0.71	0.25	0.30	16821
weighted avg	0.75	0.57	0.59	16821
samples avg	0.73	0.66	0.57	16821

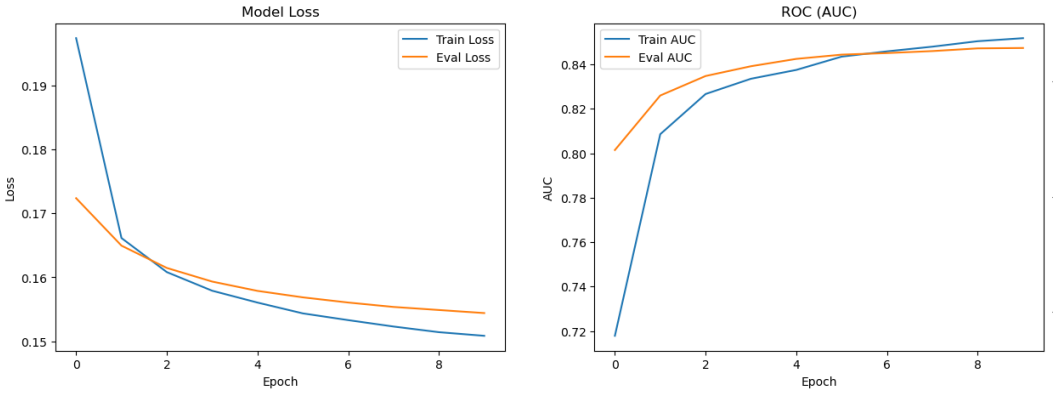

Figure 11: Dense model on sense embeddings training/evaluation graph

In this case model training didn't show any signs of overfitting after training for 10 epochs. When applied to low-dimensional sense vectors (50-dimensional), the same dense network demonstrated moderate performance, reaching a validation AUC of 0.85 but with lower practical F1-scores (macro-F1 of 0.30, micro-F1 of 0.65). Frequent categories like related, `aid_related`, and `weather_related` were still correctly predicted, but rarer categories deteriorated sharply, confirming that the compact dense representations were too compressed for complex multi-label classification task.

4.3 BiLSTM on Word2Vec Embeddings

Table 13: BiLSTM on Word2Vec embeddings

Category/Metric	Precision	Recall	F1-Score	Support
related	0.85	0.94	0.89	4018
micro avg	0.78	0.57	0.66	16821
macro avg	0.67	0.25	0.30	16821
weighted avg	0.75	0.57	0.60	16821
samples avg	0.75	0.66	0.58	16821

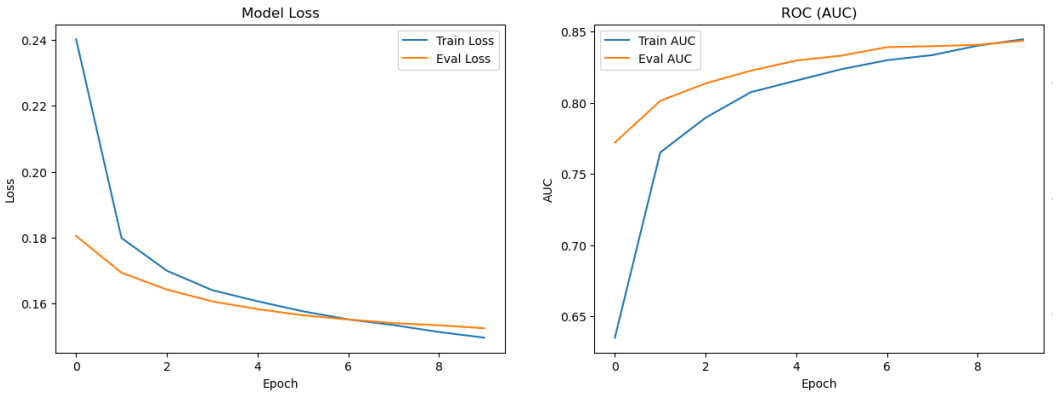

Figure 12: BiLSTM on Word2Vec embeddings training/evaluation graph

A bidirectional LSTM network using pre-trained Word2Vec embeddings achieved a validation AUC of 0.84 after 10 epochs, indicating good representational learning. However, F1-scores remained worse comparable to the dense model: macro-F1 of 0.30, micro-F1 of 0.66, and weighted-F1 of 0.60. Although the recurrent layers captured some temporal dependencies, they did not yield any improvements, largely because the majority of messages were short (often fewer than 12 tokens), limiting sequential information available for the LSTM's recurrent layers to exploit. Performance was strong for high-frequency labels (`related`, `aid_related`, `weather_related`) but inconsistent for rare categories, mirroring the imbalance observed in data analysis phase.

4.4 BiLSTM with Trainable Embeddings

Table 14: BiLSTM with trainable embeddings

Category/Metric	Precision	Recall	F1-Score	Support
related	0.87	0.90	0.89	4018
micro avg	0.76	0.56	0.64	16821
macro avg	0.66	0.22	0.24	16821
weighted avg	0.73	0.56	0.57	16821
samples avg	0.74	0.63	0.55	16821

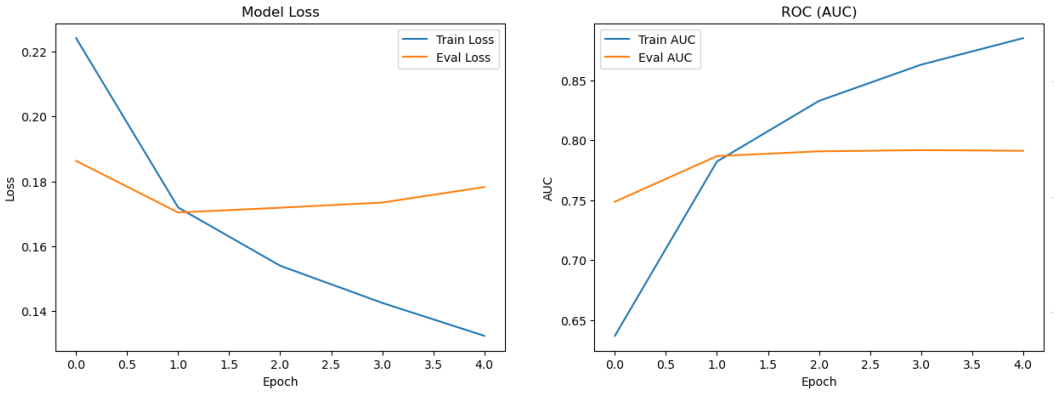

Figure 13: BiLSTM with trainable embeddings training/evaluation graph

Introducing a trainable embedding layer (vocabulary size of 8000, embedding dimension of 50) did not improve overall generalization. Despite achieving a training AUC of 0.88, validation AUC plateaued at 0.79 with rising loss, indicating overfitting after 5 epochs. The model's macro-F1 dropped to 0.24 and weighted-F1 to 0.57, with many minority classes completely unrecognized. This underperformance is attributed to the relatively small dataset and short input lengths, which hindered the model's ability to learn word embeddings from scratch. The results highlight that training embeddings without external pre-training is ineffective for low-resource, short-text datasets.

In addition we have examined the lighter text cleaning variant before tokenization, that is excluding the lemmatization and stop-word removal phase.

Table 15: Results of trainable embeddings with BiLSTM with light text preprocessing

Category/Metric	Precision	Recall	F1-Score	Support
related	0.87	0.91	0.89	4018
micro avg	0.75	0.57	0.65	16821
macro avg	0.68	0.22	0.24	16821
weighted avg	0.73	0.57	0.58	16821
samples avg	0.73	0.64	0.56	16821

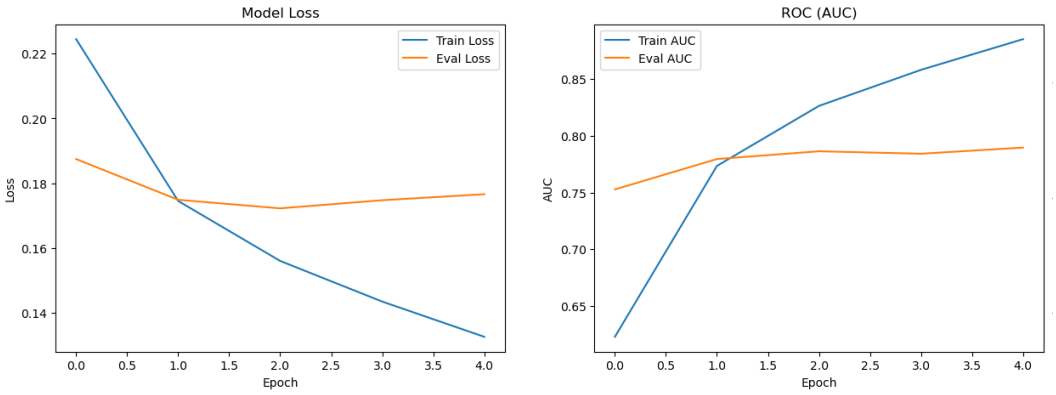

Figure 14: Trainable embeddings with BiLSTM training/evaluation graph (light text preprocessing)

No significant difference was observed.

4.5 DistilBERT Transformer

It is important to state that we have employed `distilbert-base-uncased`, which is non-case sensitive [21].

Table 16: DistilBERT results

Category/Metric	Precision	Recall	F1-Score	Support
related	0.87	0.93	0.90	4018
micro avg	0.77	0.63	0.69	16821
macro avg	0.66	0.29	0.32	16821
weighted avg	0.73	0.63	0.64	16821
samples avg	0.75	0.69	0.60	16821

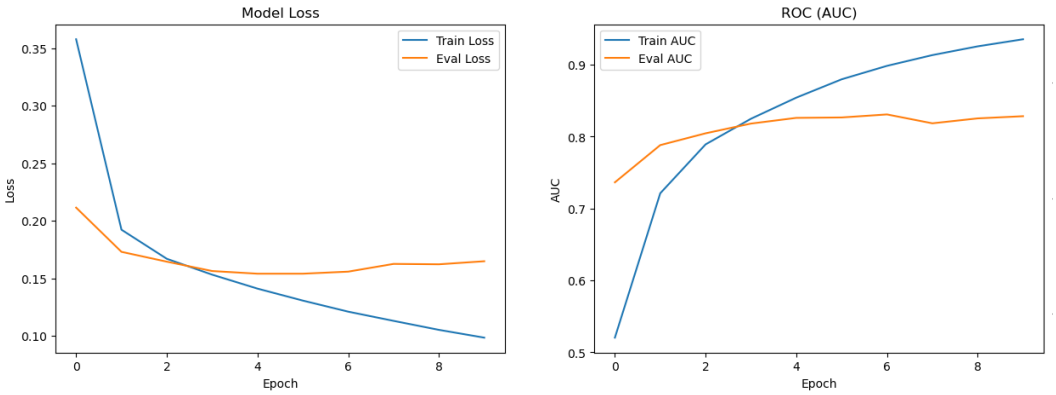

Figure 15: DistilBERT training/evaluation graph

During training, the model exhibited smooth convergence until epoch 5 and then started overfitting, which is seen from binary cross-entropy loss graph for training and evaluation samples, finishing with the validation AUC of 0.83.

The final evaluation yielded a macro-F1 of 0.32, micro-F1 of 0.69, and weighted-F1 of 0.64, indicating strong performance on dominant categories and poor generalization across others. High precision and recall were achieved for frequent categories such as `related` (F1 0.90), `aid_related` (F1 0.75), `food` (F1 0.81), and `weather_related` (F1 0.80). However, categories with limited examples such as `tools`, `hospitals`, `shops`, and `aid_centers`, were rarely detected (F1 0.00), consistent with severe class imbalance in the dataset, and short document length. Mid-frequency classes (`medical_help`, `refugees`, `transport`) achieved partial recognition with F1 scores between 0.3 and 0.5.

AUC curve and per-class confusion matrices confirmed that the model effectively captured semantic relations for distinguishing disaster related and non-related messages, although it can't properly specify any other disaster-related categories for such messages.

In addition it was decided to examine the transformer performance on the lighter text cleaning, as the transformer was pretrained on almost raw text (we have used uncased-variant), the lack of stop-words, punctuation along with lemmatization removed valuable syntactic and semantic cues, shortening inputs and reducing the richness of contextual information available during fine-tuning. That was further examined in the example below, when we have removed the step of stop-word removal and lemmatization from text preprocessing.

The sequence length in general is the key for better contextual modeling, even for state-of-art architectures it is diffucult to deduce mutliple categories from very short sequences, and it is the main problem of this dataset.

A slight improvement was noticed in comparison with previous text preprocessing variant as well as slightly more stable training graph, though it anyway started to overfit around 5th epoch.

Table 17: DistilBERT on light text cleaning

Category/Metric	**Precision**	**Recall**	**F1-Score**	**Support**
related	0.87	0.94	0.91	4018
micro avg	0.78	0.64	0.70	16821
macro avg	0.70	0.31	0.34	16821
weighted avg	0.75	0.64	0.65	16821
samples avg	0.75	0.71	0.61	16821

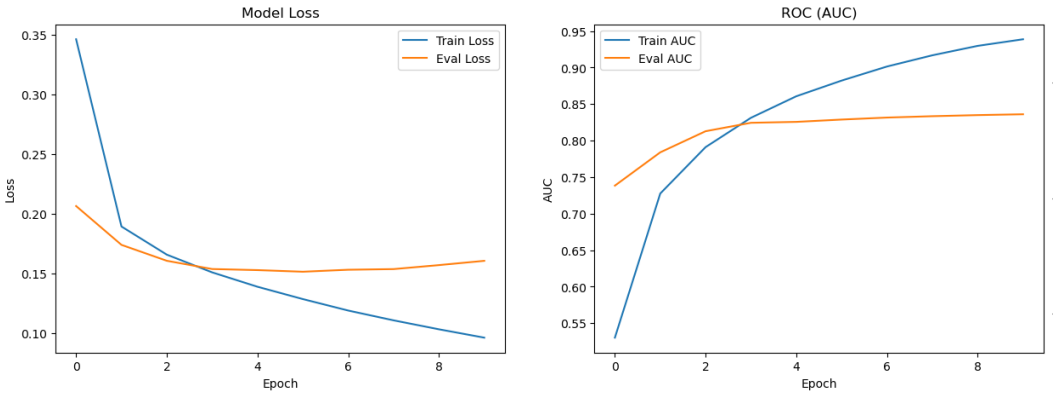

Figure 16: DistilBERT on light text cleaning training/evaluation graph

These experiments highlight the sensitivity of transformer and vector sequence models in general to text preprocessing. While classical ML models often benefit from normalization and token simplification, pretrained transformers depend on intact linguistic structure. Restoring some features improved both recall and overall F1-scores, confirming that minimal preprocessing yields better alignment with DistilBERT's learned language patterns.

5 Named Entity Recognition Results

The NER process successfully identified key entities across the corpus. A frequency analysis confirmed that the most common entity types were highly relevant to disaster response: `GPE` (Geopolitical Entities), `DATE`, and `ORG` (Organizations). Sample extractions from messages highlighted the model's ability to pinpoint locations, timings, and organizations, such as `"haiti [GPE]"`, `"tomorrow [DATE]"`, and `"radio nirvana [ORG]"`.

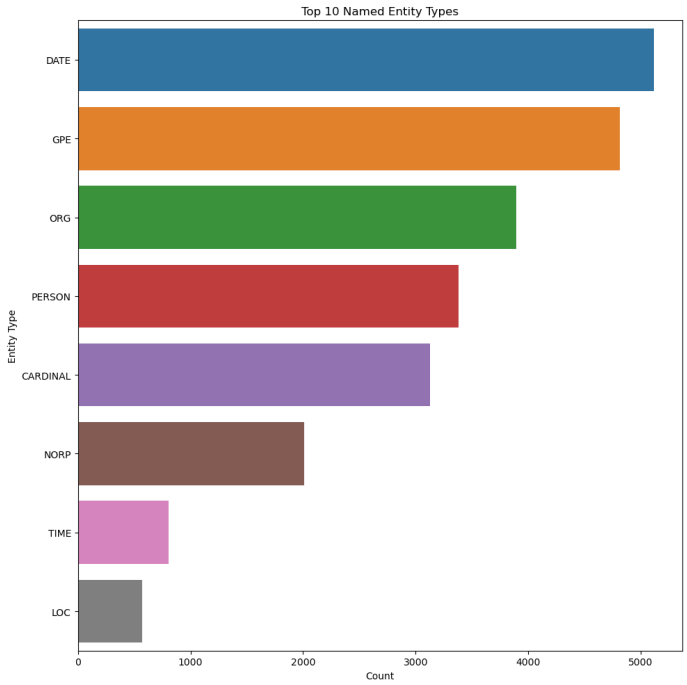

Figure 17: Top 10 Named Entity Types bar chart

Figure 18: Sample entity visualization using displaCy

The prevalence of location (`GPE`) and organization (`ORG`) entities provides direct, actionable data for emergency routing and coordination. The model did produce some noise, particularly in the `PERSON` entity type, but overall proved highly effective at extracting the most critical information types.

6 Topic Modeling Analysis

The LDA model identified coherent, broad themes such as basic needs (`water, food, supply`), flooding (`rain, river, flood`), and major disasters (`earthquake, haiti, hurricane`). In contrast, BERTopic produced more specific, event-driven clusters like Hurricane Sandy and Chile Earthquake, alongside thematic clusters like requests for help.

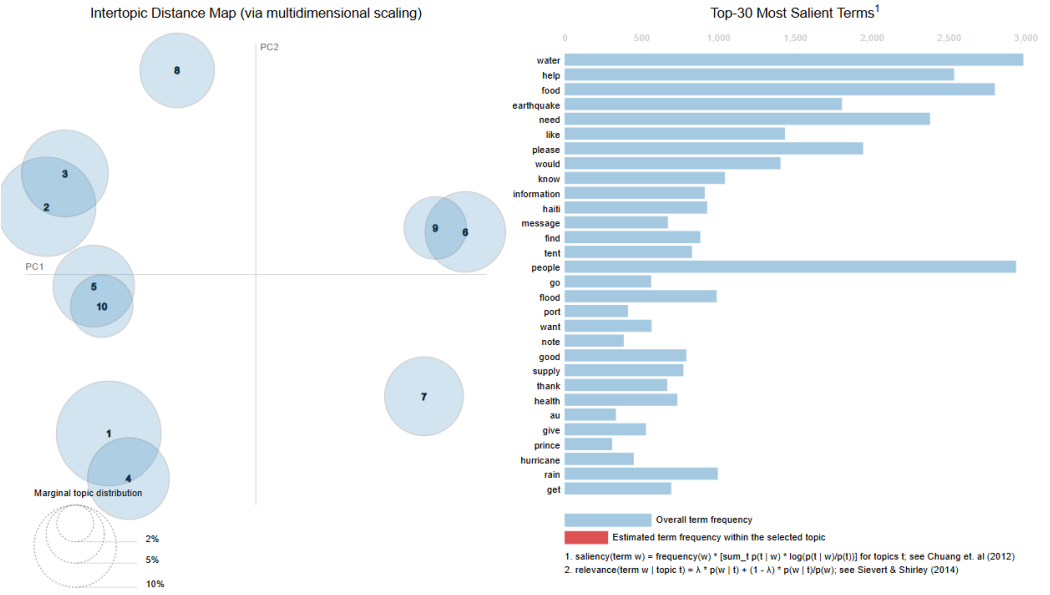

Figure 19: pyLDAvis interactive topic visualization

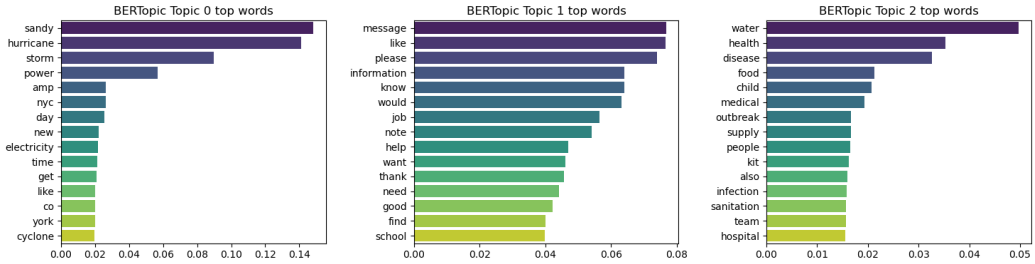

Figure 20: BERTopic bar charts (first three topics)

The two methods offer complementary value: LDA provides a high-level thematic summary of the entire corpus, which is useful for strategic planning, while BERTopic's event-specific clustering offers superior granularity for situational awareness during particular disasters.

7 Sentiment Analysis Findings

The sentiment analysis revealed a complex emotional landscape. VADER, sensitive to intense language, classified the majority of messages as positive. TextBlob, a more general-purpose tool, showed a more balanced distribution, identifying both positive and neutral sentiments. Analysis by category revealed a clear pattern: categories describing damage and loss (e.g., `death`, `military`) had strongly negative sentiment, while those related to aid and support (e.g., `offer`, `money`) had positive sentiment.

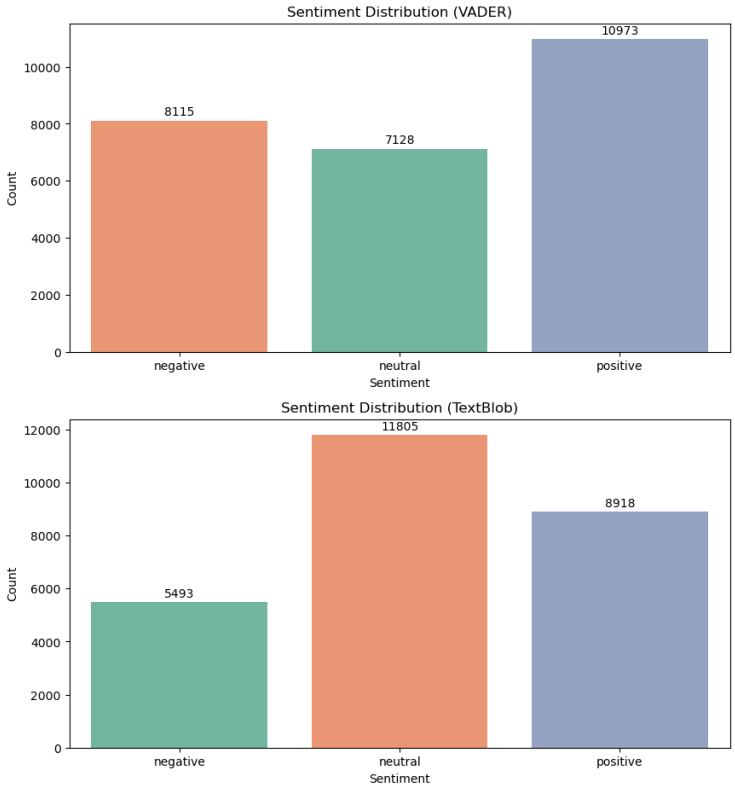

Figure 21: Sentiment distribution comparison charts

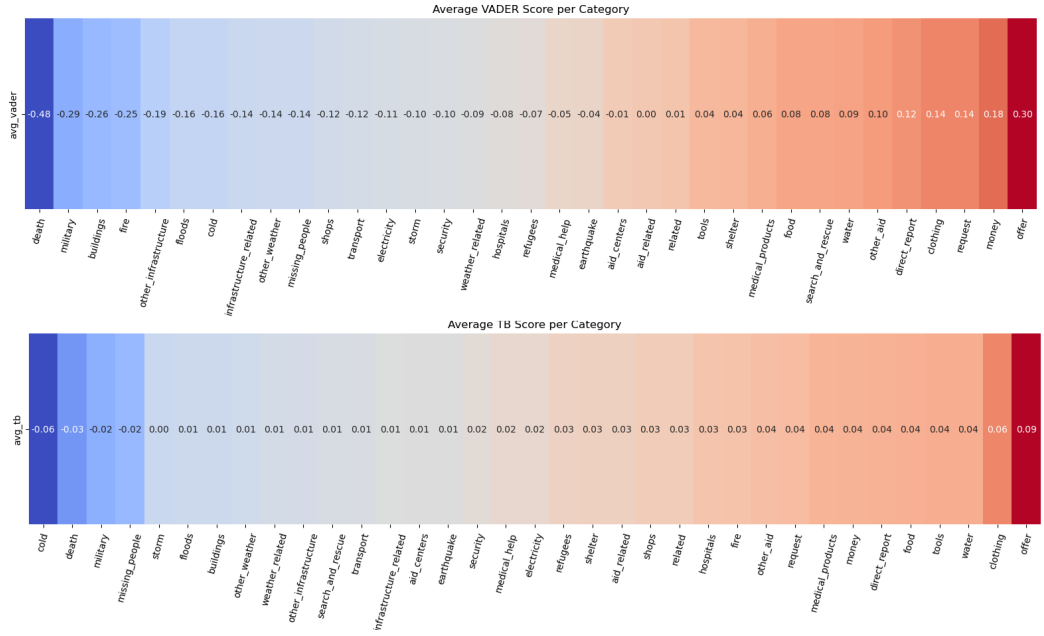

Figure 22: Heatmap of sentiment by category

Sentiment analysis shows that disaster communication tends to be neutral to slightly positive overall, driven by coordination, aid, and requests for help, with pockets of strong negativity in damage- or fatality-related categories. VADER detects sharper emotional polarity, while TextBlob tends toward moderation. Together, they show a consistent thematic divide between damage-focused (negative) and aid-focused (positive) messages. VADER is likely more accurate in capturing the urgency and distress inherent in crisis communications. The strong correlation between sentiment and category provides an additional, crucial dimension for triage, flagging messages in high-negative sentiment categories as particularly urgent.

8 Methodological Limitations

The project faced several limitations. The aggressive text cleaning, while beneficial for traditional machine learning, has removed capitalization cues that might aided Named Entity Recognition (NER). It, as well, have reduced the performance of deep learning models. Furthermore, the nature of messages of being short in length limited the amount of contextual information available for sequential models such as LSTMs and Transformers. In addition, severe multi-label class imbalance meant that models struggled to learn rare categories which is an issue that would require more advanced techniques, such as data augmentation or cost-sensitive learning, beyond the scope of this pipeline.

9 Practical Implications for Disaster Response

The implemented pipeline provides multiple actionable components for emergency response. Traditional machine learning models, such as Support Vector Machine (SVM) with BoW or TF-IDF features, offer a robust and efficient method for automatic message categorization. Named Entity Recognition (NER) enables the extraction of entities such as `GPE` (locations), `DATE`, and `ORG`, which support geographic triage and coordination efforts. Sentiment Analysis adds a layer of urgency estimation by distinguishing negative from positive emotional tone to assist in prioritization. Furthermore, Topic Modeling methods such as LDA and BERTopic summarize public concerns and emerging discussion themes, supporting strategic decision-making.

Together, these integrated tools can significantly reduce the manual burden on emergency operators, enhance real-time situational awareness, and improve the overall efficiency of information processing during disaster response scenarios.

10 Discussion

The aggressive cleaning strategy effectively prepared text for traditional machine learning by reducing noise and dimensionality. However, this approach represented a trade-off that removed structural features (punctuation, capitalization, stop-words) valuable for reccurent and transformer-based models, as well as, Named Entity Recognition (NER).

In addition, traditional feature representations effectively captured the disaster-specific lexicon, while embeddings learned semantic relationships. The vocabulary size difference between vectorizers (6,224) and Word2Vec (6,366) indicated that the embedding model captured additional morphological variants. The short message length characteristic fundamentally constrained sequential model design.

The linear dense network trained on TF-IDF features achieved the most balanced performance among deep architectures, with a macro-F1 score around 0.38 and a micro-F1 near 0.67, followed by DistilBERT. Models built on Word2Vec embeddings or trainable embedding layers underperformed, largely due to the dataset's short message lengths, small vocabulary, and class imbalance. The LSTM architecture did not provide any advantage, as most messages contained few tokens after cleaning, offering little sequential information for the recurrent layers, the same limitation also applied to transformer-based architectures such as DistilBERT. Additionally, training word embeddings from scratch either using Word2Vec or trainable embedding layers on such a small corpus led to fast overfitting and weak generalization to rare labels.

For short, noisy posts and small datasets, classical linear models over BoW and TF-IDF remain a strong baseline, outrperforming Deep Models in this study. These models offer interpretability, efficiency, and resilience to class imbalance, all of which are critical for real-world disaster response systems. While deep architectures show promise in learning semantic and contextual relations, they require larger and more diverse data to reach their potential.

Auxiliary modules, which are Named Entity Recognition (NER), Sentiment Analysis, and Topic Modeling added interpretive depth to the system. The spaCy-based NER module effectively extracted geographic (`GPE`) and organizational (`ORG`) entities, yielding structured, actionable information for identifying key locations and actors in disaster communications. Despite minor noise in `PERSON`-type entities, it demonstrated strong potential for automated tagging in real-world response pipelines. Sentiment Analysis using VADER and TextBlob introduced an emotional dimension, with VADER capturing urgency and distress particularly in `death` and `military` messages, while TextBlob produced smoother polarity trends. The alignment between sentiment and disaster categories suggests sentiment cues can help prioritize urgent communications. Topic Modeling with LDA and BERTopic uncovered latent structures within the corpus. LDA identified broad themes such as basic need and infrastructure damage, whereas BERTopic revealed specific, event-driven clusters like Hurricane Sandy and requests for help.

11 Conclusion

This project implemented a complete natural language processing pipeline for disaster response message classification, including data cleaning, feature extraction, classical and deep learning classification, as well as complementary analyses.

Across all experiments, traditional machine learning models, particularly `LinearSVC` with BoW and TF-IDF features, achieved the most consistent and generalizable results across multi-label classification of messages into 35 distinct categories, confirming that linear approaches remain highly competitive on small, imbalanced datasets. Deep architectures for vector sequence processing such as LSTM and DistilBERT didn't demonstrate any significant improvements for frequent categories and struggled with rare labels even more, mainly due to the short message length and limited contextual richness of the dataset.

Beyond performance metrics, the pipeline successfully integrated multiple analytical components into a unified framework for disaster data understanding.

1. NER effectively extracted geographical and organizational information;
2. Topic Modeling revealed thematic structures within the corpus
3. Sentiment Analysis provided insights into the emotional polarity of crisis communications.

Together, these layers produce a multi-faceted analytical toolset capable of supporting emergency response operations through automated categorization, triage, and summarization.

Future work should explore approaches for mitigating class imbalance, such as data augmentation, focal loss, or semi-supervised learning, exploring TF-IDF weighted sentence embeddings. Expanding the dataset to include multilingual and cross-platform sources would further enhance model robustness and generalization. Additionally, integrating multimodal data such as images or geolocation metadata, could substantially improve situational awareness and decision-making capacity in real-time disaster management. It is also important to state that on such a small corpus, it is better to use pre-trained embedding models.

Ultimately, this project demonstrates that even modestly scaled NLP pipelines, when designed with domain constraints in mind, can deliver significant operational value in crisis response scenarios, bridging the gap between academic modeling and practical humanitarian applications.

References

[1] Muhammad Imran et al. "Processing social media messages in mass emergency: A survey". In: *ACM Computing Surveys* 47.4 (2015), pp. 1–38.

[2] Dat Quoc Nguyen et al. "Robust classification of crisis-related data on social networks using convolutional neural networks". In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (2017).

[3] Gerard Salton and Christopher Buckley. "Term-weighting approaches in automatic text retrieval". In: *Information Processing & Management* 24.5 (1988), pp. 513–523.

[4] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2013.

[5] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[6] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[7] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of NAACL-HLT*. 2019.

[8] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108* (2019).

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[10] Maarten Grootendorst. "BERTopic: Neural topic modeling with contextual embeddings". In: *arXiv preprint arXiv:2203.05794* (2022).

[11] C J Hutto and Eric Gilbert. "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text". In: *Proceedings of ICWSM*. 2014.

[12] Steven Loria. *TextBlob: Simplified Text Processing*. `https://textblob.readthedocs.io`. 2018.

[13] Figure Eight Inc. *Disaster Response Messages Dataset*. `https://www.kaggle.com/competitions/disaster-response-messages`. Accessed: 2025-11-06. 2018.

[14] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009.

[15] George A. Miller. "WordNet: A Lexical Database for English". In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[16] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[17] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine Learning* 20.3 (1995), pp. 273–297.

[18] Leo Breiman. “Random forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32.

[19] Diederik P Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations (ICLR)* (2015).

[20] Explosion AI. *spaCy: Industrial-strength Natural Language Processing in Python*. `https://spacy.io/`. 2023.

[21] Hugging Face. *distilbert/distilbert-base-uncased Model Card*. `https://huggingface.co/distilbert/distilbert-base-uncased`. Accessed: 2025-11-07.