Predicting Natural Disaster Tweets Using Binary Classification

Group 8:
Tammy Dahl
Tommy Kieu
Kyle Krick
Roger Reinhardt
Jordan Spencer

Abstract

This paper implements and evaluates binary classification machine learning models to predict whether a given tweet describes a natural disaster. We applied traditional machine learning techniques alongside modern embedding methods such as TF-IDF, Word2Vec, and BERT embeddings. Our best-performing model achieved a training F1 score of approximately 0.93, demonstrating the effectiveness of deep learning and ensemble approaches over traditional machine learning methods for this type of classification task.

Introduction and research problem

Social media platforms, particularly Twitter, have become critical communication tools during natural disasters. Individuals often share real-time observations and reports, making it possible to detect emergencies earlier and respond more effectively. However, automatically identifying whether a tweet describes a real natural disaster remains a difficult challenge due to the short, informal, and often ambiguous nature of tweets, including the use of slang, sarcasm, and metaphors.

Our project attempts to address this problem by implementing and evaluating binary classification machine learning (ML) models to predict whether a given tweet is describing a real natural disaster. This work applies fundamental machine learning and natural language processing (NLP) techniques, including text preprocessing, vectorization, embedding and model training with both traditional and deep learning methods. By analyzing and comparing different models, including Support Vector Machine, Logistic Regression, Random Forest, and BERT-based neural networks, we aim to develop a reliable system for disaster response agencies and public safety efforts.

Furthermore, this project allows us to develop critical skills in NLP. Given recent breakthroughs in NLP, namely large language models (LLM) like ChatGPT and LLaMA, we anticipate that we will need NLP and machine learning skills in our careers. This project is one step that will set us up for success whether we are training the next generation of these models or simply learning to apply them to new problems.

Related work

To inform our work on this project, we conducted a brief literature review to understand the context of this problem space. Several of the papers indicated that neural networks, specifically transformer based models like BERT are superior at natural language processing tasks.

Duraisamy and Natarajan [1] explored the use of different deep learning models for disaster prediction on Twitter, highlighting the effectiveness of models like CNNs and RNNs for capturing complex text patterns.

Dharro et al. [2] proposed a disaster tweet classification method that combined BERT embeddings with optimized convolutional neural networks (CNNs), demonstrating improved classification performance compared to traditional techniques.

Ma [3] conducted experiments using BERT for disaster tweet classification, finding that transformer-based models significantly outperformed traditional methods like logistic regression in capturing context and improving accuracy.

Fontalis et al. [4] performed a comparative study of deep learning approaches for detecting and classifying natural disasters from social media posts, concluding that deep learning models generally outperform classical machine learning models for complex, unstructured text data.

While we had initially planned to implement and evaluate just Support Vector Machine and Logistic Regression models, we realized from reading these related works that we would likely achieve better results with BERT and other deep learning-based models. With this context established, we set out to train our own machine learning models to tackle this challenge of binary classification of natural language.

Methodology and technical details

For collaboration on this project, we used the SDSU instructional JupyterHub and the PyTorch Notebook so that we were all able to work from the same environment. This solved a lot of potential collaboration problems for us such as differing versions of

operating systems, languages and libraries. Jupyter notebooks are extremely valuable for collaborative scholarly and research related work like this project.

With our computational environment selected, we began working with the data that we would use to train our models. We leveraged data from the Kaggle competition “Natural Language Processing with Disaster Tweets” to train our models [5]. The training dataset contained the text from 10,000 tweets each labeled with a 1 for a disaster related tweet or 0 otherwise. We automated the download and organization of the raw data with a bash script and Jupyter notebook thus ensuring that we were each able to reproducibly download the raw data. We did exclude the raw data from our repository so that we did not redistribute the competition data without permission. Those who wish to verify our work from start to finish will need to create a Kaggle account and join the competition in order to gain access to the raw data.

After we had retrieved the raw data from Kaggle, we began preprocessing the data. Preprocessing included lowercasing text, removing special characters, mentions, hashtags, and applying tokenization. We experimented with multiple methods for converting text to numerical features, including Word2Vec embeddings, TF-IDF vectorization, and pre-trained BERT embeddings using distilBERT. We saved each of the preprocessed datasets for later use when we trained and evaluated our models. We believe this will be convenient for reproducing our models.

With our data preprocessed we began training machine learning models. We trained and evaluated several machine learning models including:

- Logistic Regression
- Support Vector Machine
- Random Forest
- XGBoost
- Neural Networks using BERT embeddings
- A meta-classifier modular ensemble combining predictions from multiple models

For the more traditional machine learning models like Logistic Regression, Support Vector Machine and Random Forest, we leveraged the popular scikit-learn library. This allowed us to iterate from one model to the next very quickly, leveraging the object oriented design of scikit-learn. This approach was much more time efficient than attempting to implement these machine learning models from scratch. This also freed up our time from developing, and no doubt debugging, to allow us to focus on comparing model performance.

We also experimented with XGBoost which has been a popular choice for classification tasks in other data science competitions. While we didn't learn about these models in this course, we learned enough on our own to be able to train and apply it to our binary classification task.

In our initial proposal, we had omitted more advanced machine learning models such as deep learning-based models. At the time of submitting the proposal we had not yet learned about more advanced architectures. However, after reading about other work in this problem space, and with the encouragement of our professor from her feedback on our proposal, we also experimented with BERT. This model uses the transformers architecture which uses self-attention layers and is very adept at NLP tasks.

Training was performed using K-Fold cross-validation and different models were compared based on their validation performance as measured by their F1 scores. Support Vector Machine was initially tested and showed weaker performance with Word2Vec embeddings. After adjustments to the embedding strategy, Support Vector Machine performance improved notably when paired with the TF-IDF features rather than Word2Vec.

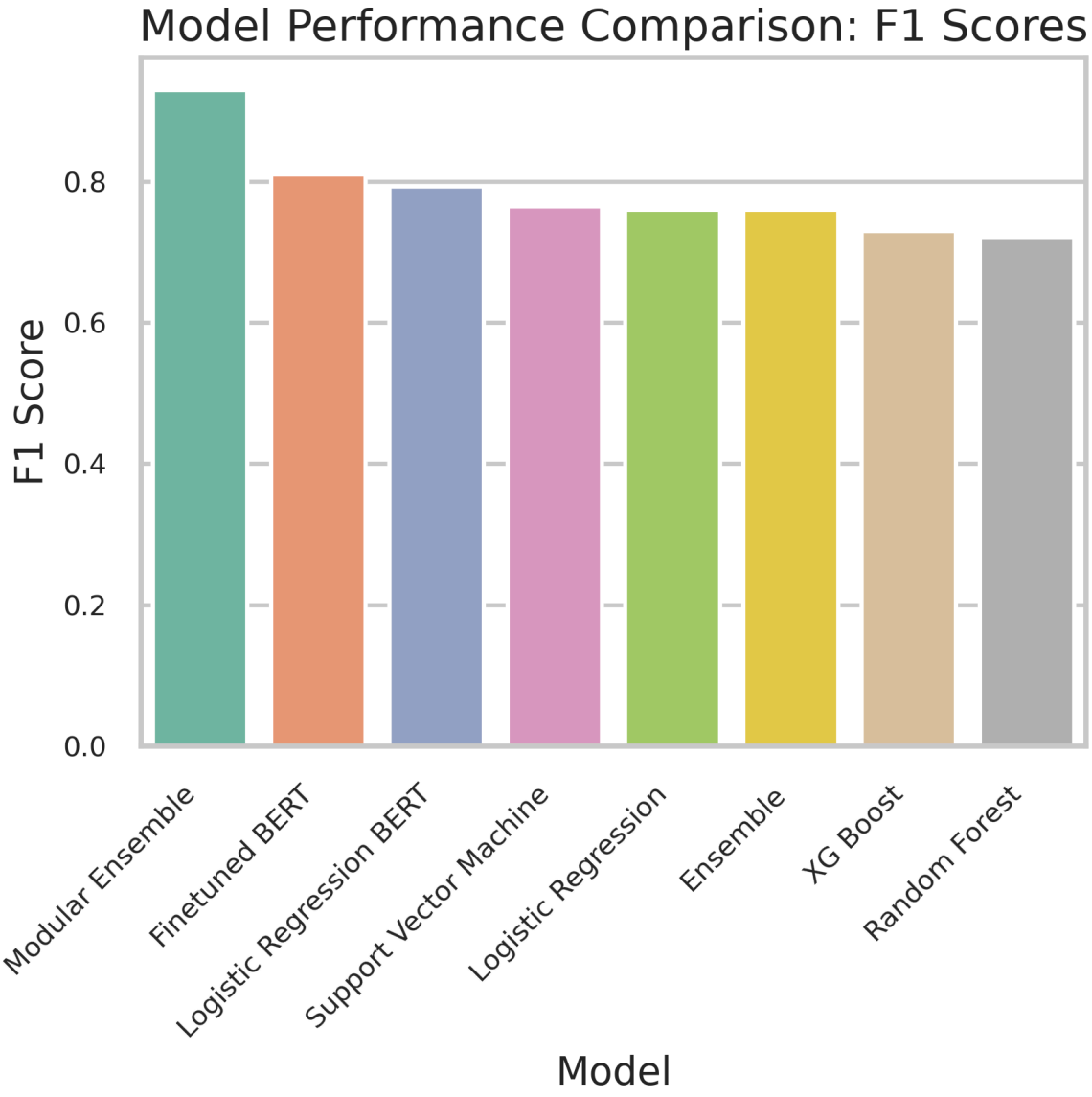
As we trained each model, we submitted its predictions for the test data for the Kaggle competition and iteratively improved our score and our placement on the leaderboard. Eventually, we decided to use an ensemble approach which works by combining several models to synergize their predictions. This resulted in superior performance compared to our individual models.

Retraining all models on the complete training set before final testing further enhanced Support Vector Machine's F1 score. Despite these improvements, Support Vector Machine did not outperform models based on BERT embeddings or Random Forest. These observations highlight the sensitivity of traditional models like Support Vector Machine to feature engineering choices in natural language processing tasks.

Experimental results and analysis

For evaluating our models, we used the confusion matrix and associated scores for accuracy, precision, recall and F1 score. These were a perfect fit for evaluating our model since it was a binary classification problem. Initially we were primarily focused on each of our model's accuracy as our key performance indicator, since accuracy reflects correct predictions among all examples.

However, once we had started submitting our models' test predictions to the Kaggle competition, we realized that they were scoring according to F1 scores. Once we realized the metric that we were being ranked by, we switched to F1 scores as our key performance indicator. Below is a summary of all of our models' performance according to their F1 scores.



Temporarily ignoring the ensemble models, our best individual model results came from the Finetuned BERT and Logistic Regression BERT, logistic regression using BERT embeddings, models achieving F1 scores of 0.81 and 0.79 respectively. Our poorest performing model, Random Forest, showed signs of overfitting, where training loss was decreasing, but validation loss began to increase. These results show that BERT

embeddings combined with neural networks provided more stable results. This suggests that for this specific task traditional machine learning approaches like Logistic Regression are still effective, however deep neural networks like BERT offer superior performance.

The BERT-based neural network model ultimately performed best overall, achieving an F1 score of approximately 0.81 on the validation data. This model likely benefited from the high quality feature representation provided by the distilBERT embeddings, as well as the robust transformers architecture which is well adapted to natural language tasks.

Cross-validation helped ensure that our results were stable across different subsets of data. Our findings also align with recent research suggesting that deep learning models, particularly transformer-based models, outperform traditional machine learning approaches to natural language processing tasks where text patterns are subtle and varied.

Ultimately, we created a meta-classifier model we called “Modular Ensemble” by combining the individual models in an effort to boost our overall performance. With this ensemble assembled, we re-trained the ensemble on the entire training dataset. This method of combining models yielded an F1 score of 0.93 on the complete training dataset. Clearly, the ensemble model leveraged the strengths of different models and improved overall performance.

Conclusion and future work

We trained and evaluated several machine learning models to predict whether or not tweets were describing natural disasters. Our final model, referred to as the meta-classifier ensemble, combined each of the models we had trained, excluding Support Vector Machine, and averaged their predictions to make a single, final prediction. This approach resulted in an F1 score of 0.93 on the test dataset. We submitted this model to the Kaggle competition earning a score of 0.83849 and placing at 76 on the leaderboard. This suggests that our modular ensemble had some difficulty in generalizing to this unseen dataset, but overall performed better than any of our individual models. Having broken the top 100 of the leaderboard, we were satisfied with the results.

We can see several paths to continuing this work through integration with novel datasets and different types of machine learning models. For example, these models could be incorporated into a workflow that analyzes social media posts in near real-time. These models could also be expanded into multiple classification models to predict the type of natural disaster being depicted in the posts. In addition to text data, social media

platforms have location, image, video and audio data all of which could be processed by various machine learning models to analyze, predict and classify natural disasters.

Our work is freely available on GitHub at the following repository link: https://github.com/KyKri/tweet-classifier/tree/main. The repository readme file should guide users through reproducing our work from start to finish. With the exception of the considerably sizable finetuned BERT model, we have also included our preprocessed datasets, model files and a prediction notebook for testing and verification of our work. We will submit a compressed version of finetuned BERT with this assignment.

Contributions of each group member

Jordan Spencer

- Served as team lead for model development and integration
- Led preprocessing pipelines, implemented TF-IDF, Word2Vec, and BERT embeddings
- Built and trained multiple models including Random Forest, XGBoost, and BERT-based neural networks, and constructed the final meta-classifier ensemble

Roger Reinhardt

- Contributed to the evaluation strategy by selecting key performance metrics (accuracy, precision, recall, F1 score)
- Assisted in reviewing and interpreting model performance results

Kyle Krick

- Focused on model development, assisting in training traditional machine learning models such as Support Vector Machines (Support Vector Machine), Logistic Regression, and Random Forest
- Contributed to tuning model hyperparameters and validating early model versions
- Contributed to the reproducibility of the repository
- Contributed to the prediction notebook allowing others to test our models

Tammy Dahl

- Contributed to writing and organizing the final project report, including synthesizing technical content and ensuring alignment with project objectives
- Researched and planned approaches for visualizing model predictions to support the final presentation of results

Tommy Kieu

- Supported research efforts by identifying and sourcing references for the literature review, helping to establish the academic context for the project
- Assisted in reviewing and validating the data preprocessing workflow to ensure the dataset was properly prepared for model training

References

Duraisamy, Premkumar, and Yuvaraj Natarajan. “Twitter Disaster Prediction Using Different Deep Learning Models - SN Computer Science.” *SpringerLink*, Springer Nature Singapore, 10 Jan. 2024, link.springer.com/article/10.1007/s42979-023-02520-7.

Dharrao, Deepak, et al. “An Efficient Method for Disaster Tweets Classification Using Gradient-Based Optimized Convolutional Neural Networks with Bert Embeddings.” *MethodsX*, Elsevier, 3 July 2024, www.sciencedirect.com/science/article/pii/S2215016124002966?via%3Dihub.

Ma, Guoqin. *Tweets Classification with Bert in the Field of Disaster ...*, web.stanford.edu/class/archive/cs/cs224n/cs224n.1204/reports/custom/15785631.pdf. Accessed 27 Apr. 2025.

Fontalis, Spyros, et al. “A Comparative Study of Deep Learning Methods for the Detection and Classification of Natural Disasters from Social Media.” *SciTePress*, 21 Mar. 2023, www.scitepress.org/Link.aspx?doi=10.5220%2F0011666500003411.

Addison Howard, Devrishi, Phil Culliton, and Yufeng Guo. Natural Language Processing with Disaster Tweets. https://kaggle.com/competitions/nlp-getting-started, 2019. Kaggle.