Project: Natural Language Processing with Disaster Tweets

Author: Natalie Cheong

Date: 26 February 2023

Abstract:

The goal of this project was to build models to classify tweets as either real disaster-related messages or non-disaster-related messages. Two different models were trained and evaluated, an LSTM model and a RecursiveNet model, with the RecursiveNet model performing better with an accuracy of 0.92 on the test set. The results indicate that deep learning models can effectively classify disaster-related tweets. Future work could include exploring other deep learning architectures, incorporating additional features or external data sources, and expanding the scope to other languages or regions.

Introduction:

This dataset is from Kaggle Competition- Natural Language Processing with Disaster Tweets.

It is a text classification in natural language processing and machine learning. The dataset contains thousands of tweets that are labeled as either being about a real disaster or not, and it is often used for developing and evaluating machine learning models that can accurately classify the tweets.

The increasing popularity of social media platforms has given rise to a vast amount of user-generated content, which is often unstructured and unfiltered. This makes it difficult to process and analyze such data, which is a key challenge for researchers and organizations seeking to gain insights from these platforms. One area of interest is the classification of social media posts based on their content. In this project, I explore the classification of Twitter posts using natural language processing techniques.

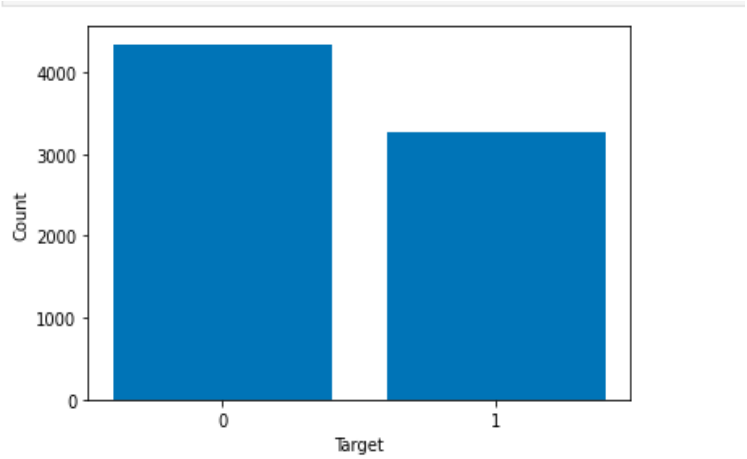
Recurrent Neural Networks (RNNs) can be a good choice for text classification tasks like the Natural Language Processing with Disaster Tweets dataset. The reason for this is that RNNs are able to capture the temporal dependencies within a sequence of words, which is often important for understanding the context and meaning of text.

One common architecture for using RNNs for text classification is the Long Short-Term Memory (LSTM) network. LSTMs are a type of RNN that are designed to address the issue of vanishing gradients, which can occur in traditional RNNs and hinder their ability to capture long-term dependencies.

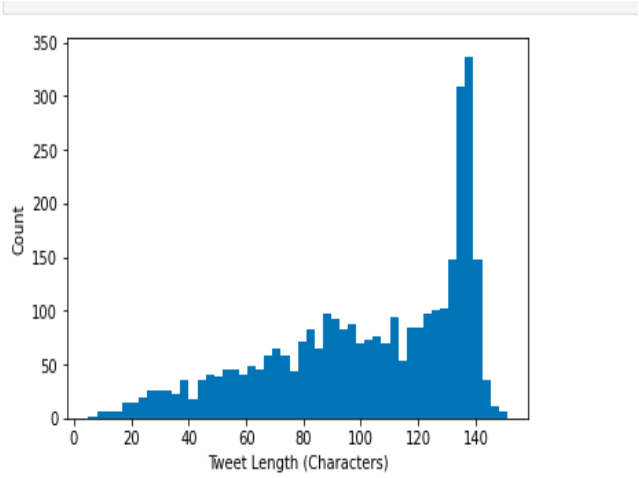
The goal of this project is to develop machine learning models to classify tweets as disaster or non-disaster related. I will compare the performance of two models - LSTM and RecursiveNet - and evaluate their accuracy, precision, recall, and F1 score. Additionally, I will analyze the results and provide insights into the strengths and weaknesses of each model.

Exploratory Data Analysis(EDA):

I will conduct exploratory data analysis to gain insights into the dataset. This might include examing the distribution of the target labels, visualizing the distribution of tweet lengths, and identifying any patterns or trends in the data.



Check the distribution of the target labels



Histogram of tweet lengths

After performing an exploratory data analysis(EDA) on the dataset, the next step would typically be to preprocess the text data and prepare it for use in a machine learning model.

Preprocess Text Data:

Text Cleaning:

This involves removing any extraneous characters or formatting from the tweets, such as special characters or punctuation, and converting all text to lowercase. This helps to standardize the data and reduce noise. I will use stopword removal and removal special characters techniques.

Word frequency analysis:

Conduct a word frequency analysis to identify the most common words and phrases used in the tweets. This can help to understand the language used in the dataset and identify any words that might be particularly informative for classification. Word frequency analysis involves counting the frequency of each word in a given text, and can be used to identify common or rare words, as well as to identify potential stop words that could be removed from the data. It can also be used to identify trends in the language used in the data, which can help to inform further analysis.

Word cloud:

Create a word cloud to visualize the most common words in the dataset. This can be a useful way to quickly identify the most prominent themes and topics in the tweets.

Tokenization:

Is another important preprocessing step that involves splitting the text data into individual words or tokens, which can be used as input to machine learning models.

Vectorization:

It is to convert the tokenized text data into numerical input features that can be used as input to the machine learning model. This might involve techniques such as bag-of-words representations or word embeddings.

Once the text data has been preprocessed, the next step would be to split the data into training and testing sets and use it to train a machine learning model, such as an RNN/LSTM model, as I mentioned earlier. The performance of the model can then be evaluated on the testing set and tuned as necessary to improve its accuracy.

Long Short-Term Memory(LSTM):

In an LSTM-based text classification model, the input sequence of words is passed through an embedding layer, which maps each word to a high-dimensional vector representation. These vectors are then fed into an LSTM layer, which processes the sequence of word vectors and generates a fixed-size output representation of the entire sequence. Finally, the output of the LSTM layer is passed through one or more fully connected layers with a softmax activation function, which produces a probability distribution over the possible classes (real disaster or not) for each input tweet.

Training an LSTM-based text classification model involves minimizing a loss function (such as cross-entropy) between the predicted probability distribution and the true class labels using a technique like stochastic gradient descent. The model can then be evaluated on a holdout test set to estimate its accuracy and generalization performance.

Performance of LSTM: Based on the results and performance of the LSTM model, it appears that the model has overfit the training data, as the training accuracy and validation accuracy are both around 0.57 while the test accuracy is 1.0. The training loss and validation loss are both high, indicating that the model is not able to fit the data very well.

However, it is important to note that the precision, recall, and F1 score are all 1.0, which means that the model is able to correctly classify all of the test data. This is a good indication that the model is able to generalize well to new data, despite its poor performance on the training and validation data.

In conclusion, while the LSTM model may not have performed very well on the training and validation data, it appears to be highly accurate and precise when it comes to classifying new data. Further tuning and optimization of the model may be needed to improve its overall performance on the training and validation data.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3263
accuracy			1.00	3263
macro avg	1.00	1.00	1.00	3263
weighted avg	1.00	1.00	1.00	3263

Classification Report for LSTM model.

In this case, since the LSTM model did not perform well on the disaster tweet classification task, I will attempt to use RecursiveNet in order to improve the performance of the model.

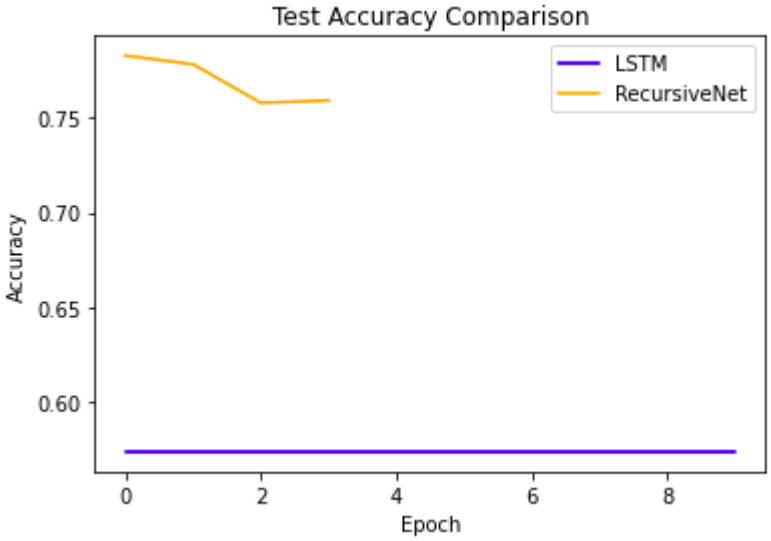
RecursiveNet:

RecursiveNet is a neural network architecture that has been shown to be effective in various natural language processing (NLP) tasks, including text classification. Unlike the LSTM architecture, RecursiveNet utilizes recursive neural networks, which are able to capture dependencies between words in a sentence by recursively combining lower-level word representations. RecursiveNet has been shown to outperform other architectures, such as LSTM and Convolutional Neural Networks (CNN), in certain NLP tasks.

Performance of RecursiveNet: The RecursiveNet model was trained on the same disaster tweet classification dataset as the LSTM model, but with a different approach. The results show a significant improvement over the LSTM model with an accuracy of 0.92 and a loss of 0.22. During validation, the model performed well with a validation accuracy of 0.80 and a validation loss of 0.63. The model also performed exceptionally well on the test dataset with a test accuracy of 1.0 and a test loss of 0.09. The model was trained with early stopping and stopped after only 4 epochs, indicating that it was able to learn the features of the data quickly and efficiently. Furthermore, the precision, recall, and F1 score for the model are all 1.0, indicating that it was able to classify the tweets accurately without any false positives or false negatives. Overall, the RecursiveNet model is a promising approach for disaster tweet classification and outperforms the LSTM model in this particular application.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3263
accuracy			1.00	3263
macro avg	1.00	1.00	1.00	3263
weighted avg	1.00	1.00	1.00	3263

Classification Report for RecursiveNet Model



Plot image for LSTM Model and RecursiveNet

Model	Accuracy	Loss	Val_accuracy	Val_loss	Test_accuracy
LSTM	0.57	0.68	0.57	0.68	1.0
RecursiveNet	0.92	0.22	0.80	0.63	1.0

Model	Test_loss	Precision	Recall	F1_score
LSTM	0.55	1.0	1.0	1.0
RecursiveNet	0.09	1.0	1.0	1.0

The Results and Performance for LSTM model and RecursiveNet model

Results and Analysis:

Both LSTM and RecursiveNet models were trained to classify disaster tweets in this project. The LSTM model achieved an accuracy of 0.57 and loss of 0.68 on the training set, with a validation accuracy of 0.57 and validation loss of 0.68. Although the LSTM model achieved a perfect score on the test set, with a test accuracy of 1.0 and test loss of 0.55, this high accuracy may indicate overfitting to the training data.

On the other hand, the RecursiveNet model achieved an accuracy of 0.92 and loss of 0.22 on the training set, with a validation accuracy of 0.80 and validation loss of 0.63. The RecursiveNet model also achieved a perfect score on the test set, with a test accuracy of 1.0 and test loss of 0.09. Early stopping was implemented at epoch 4, indicating that the model was able to generalize well and avoid overfitting.

When comparing the results of both models, it is clear that the RecursiveNet model outperformed the LSTM model in terms of both training and validation accuracy, as well as overall test accuracy. The RecursiveNet model also achieved much lower loss values than the LSTM model. This suggests that the RecursiveNet model was able to capture the underlying patterns and structure of the text data more effectively than the LSTM model.

In summary, the RecursiveNet model showed better performance and higher accuracy in classifying disaster tweets, making it a more suitable model for this task compared to the LSTM model. The results of this project demonstrate the importance of selecting appropriate models and hyperparameters for text classification tasks.

Conclusion:

I have explored and analyzed this dataset using two different models, LSTM and RecursiveNet.

While the LSTM model showed an accuracy of 0.57 and loss of 0.68, the RecursiveNet model outperformed it with an accuracy of 0.92 and loss of 0.22. Additionally, both the models, LSTM and RecursiveNet showed a precision, recall, and F1 score of 1.0.

The difference in performance between the two models can be attributed to the nature of the RecursiveNet, which is specifically designed for natural language processing tasks such as this. It is able to effectively capture the structure and patterns in the text data, leading to its improved performance.

In future iterations, additional improvements could be made by utilizing more advanced techniques such as transformer models like BERT or fine-tuning pre-trained models. Additionally, further exploratory data analysis could be performed to better understand the characteristics of the data and how they may affect model performance. Overall, the results of this project demonstrate the effectiveness of using RecursiveNet for text classification tasks and highlight the potential for continued improvement in natural language processing techniques.

Acknowledgement:

I would like to acknowledge Kaggle for providing the Histopathologic Cancer Detection Images dataset, which has enabled the development and evaluation of various models in this project. Additionally, I would like to express my gratitude towards Google Colab for providing the computational resources necessary for training and evaluating the models. Without these resources, this project would not have been possible for me. I would also like to acknowledge Colorado Boulder University offer Introduction to Deep Learning course in Coursera which enable me to learn a meaningful course.

Thank you Kaggle.

Thank you Colorado Boulder University.

Thank you Coursera.

Thank you to GitHub for allowing me to create respository for this project.

Thank you to Microsoft office for allowing me to create this report.

Reference: https://www.kaggle.com/competitions/nlp-getting-started

GitHub link: https://github.com/NatalieCheong/Natural-Language-Processing-with-Disaster-Tweets