Natural Language Processing with Disaster Tweets (Kaggle)

Ritesh Kumar

2024WI_MS_DSP_422-DL_SEC61: Practical Machine Learning

Module 8 Assignment

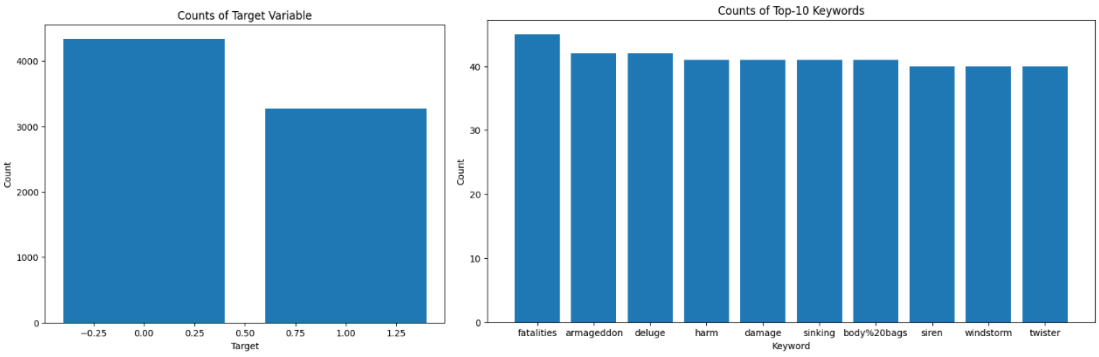
Natural Language Processing with Disaster Tweets

Donald Wedding and Narayana Darapaneni

March 5, 2024

Exploration

Our exploratory data analysis (EDA) revealed that the dataset appears to be clean and well-structured, with no duplicate rows present, indicating good data hygiene and potentially streamlined preprocessing stages. It contains 221 unique keywords, suggesting a diverse set of terms that may be relevant for analysis or feature engineering. The target variable, which is likely binary, has a slightly imbalanced distribution with 4,342 instances of the negative class (labeled as '0') and 3,271 instances of the positive class (labeled as '1'). This imbalance should be taken into consideration during the modeling phase to ensure the model does not bias towards the more frequent class.



To prepare text data for modeling, we convert text to lowercase and remove punctuation to standardize inputs. We then set up a vectorization layer to convert text into sequences of integers, using a vocabulary size of 4096 to balance between memory use and capturing diverse words, and a sequence length of 10, assuming the most relevant context for tasks like classification or tagging can be captured in this brief span, making data representation efficient and focused.

Next, we advanced the preparation of our training data by implementing a methodology that involved the creation of skip-grams with negative sampling. This process was meticulously carried out by iterating over sequences of text, adhering to predetermined parameters such as

a specified vocabulary size, window size for identifying context words, and a set number of negative samples. Through this approach, we were able to generate comprehensive lists that included target words, their respective contextual words, and corresponding labels. This methodological framework was specifically designed to aid in the development of models capable of understanding and capturing the intricate semantic relationships between words, thereby enhancing their ability to accurately interpret and process natural language. By focusing on both the immediate context of words and incorporating negative examples, our strategy aimed to refine the model's sensitivity to word usage patterns, ultimately contributing to a more nuanced and effective representation of language semantics within the model's architecture.

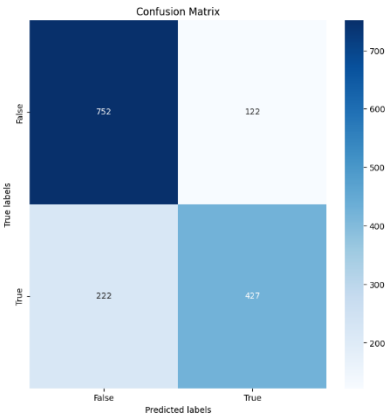
The code snippet detailed the preparatory stages for a text classification task within a machine learning framework. Initially, it extracted features (X) and labels (y) from the dataframe. Subsequently, this data was divided into training and validation sets. A tokenizer was then established and applied to the training text, transforming words into numerical sequences. These sequences underwent padding to standardize their length, rendering them compatible with neural network model inputs. This preparation aimed to aptly organize the text data for both training and model performance evaluation purposes. The sizes of the training and validation datasets were 6090 and 1523, respectively, with the maximum sequence length set to 60.

Next, we developed a model utilizing a sequential approach with various layers, including an Input layer with a shape of (60,), an Embedding layer with a dynamically chosen output dimension, and three Bidirectional LSTM layers with varying units and regularization. We incorporated Dropout for regularization and a Dense layer for output with a sigmoid activation function. The model was compiled using the Adam optimizer, with the learning

rate chosen through hyperparameter tuning. We employed a RandomSearch tuner to identify the best hyperparameters over 10 trials, each with 3 executions, using validation accuracy as the objective. The search involved training on padded training data and validation data, with early stopping based on validation loss to prevent overfitting. After determining the best hyperparameters, we trained the model for an optimal number of epochs, as identified by the highest validation accuracy achieved during training. The training process revealed insights into the model's performance, with the best epoch determined based on validation accuracy.

The final step involved retraining the model up to the best epoch and then evaluating it on the validation set. The evaluation showed significant validation accuracy, indicating the model's effectiveness in classifying the text data. The detailed logs of training and validation phases, including loss and accuracy metrics, provided a comprehensive overview of the model's learning trajectory and its performance at various stages.

The output log illustrated the neural network's training journey across 10 epochs, where it achieved a high training accuracy of up to 99.56% by the final epoch. Despite the high training accuracy, validation accuracy peaked at 78.27% during the 9th epoch and then slightly declined, with a gradual increase in validation loss observed. This pattern suggested that the model was overfitting, indicating it performed exceptionally well on the training data but failed to generalize effectively to new, unseen data.



The confusion matrix summarizes the performance of a classification model on the validation set. The model correctly predicted 'True' labels 427 times and 'False' labels 752 times. However, it incorrectly predicted 122 'True' labels as 'False' (false negatives) and 222 'False' labels as 'True' (false positives).

```
48/48 [==============================] - 0s 7ms/step
Precision: 0.7777777777777778
Recall: 0.6579352850539292
F1 Score: 0.7128547579298832
ROC-AUC: 0.8314437278968172
MCC: 0.5338670999270845
Cohen's Kappa: 0.528835777083903
```

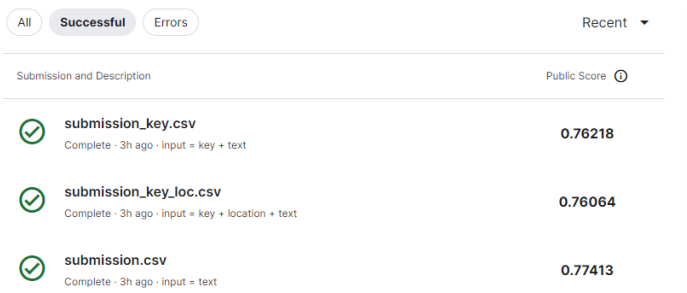
A precision of 0.778 indicated that, when the model predicted the positive class, it was correct approximately 77.8% of the time. The recall of 0.658 showed that the model successfully identified 65.8% of all actual positives. The F1 score of 0.713 reflected a reasonably good balance between precision and recall that the model achieved. The ROC-AUC score of 0.831 denoted a very good ability of the model to discriminate between the positive and negative classes. Lastly, the Matthews Correlation Coefficient (MCC) of 0.534 and Cohen's Kappa of 0.529 confirmed that the model had substantial predictive quality, significantly surpassing random chance, as both measures take into account true negatives and the balance of the dataset, unlike other metrics such as accuracy.

The conclusion for this model, based on the observed metrics and its performance on Kaggle test data, indicated that it had a commendable ability to accurately classify the given texts, achieving a respectable Kaggle accuracy score of 0.77413. However, the training logs suggested potential overfitting, as the model achieved much higher accuracy on the training data than on the validation set. The model's precision, recall, and F1 scores pointed towards an effectively balanced ability to correctly predict the positive class and cover a majority of the positive samples. The ROC-AUC score of 0.831 suggested a strong discriminatory capacity between the classes. The Matthews Correlation Coefficient and Cohen's Kappa scores reinforced the meaningfulness of the model's predictions, beyond what could be

attributed to class imbalance or random chance. To improve this model, more regularization could be applied to address overfitting, along with better hyperparameter tuning, or employing ensemble methods. Exploring additional data, conducting feature engineering, or adopting more advanced model architectures might also enhance its generalization capabilities, thereby boosting its performance on unseen data. In summary, the model showed promise but necessitated additional refinement to ensure robust performance in real-world applications.

We experimented with three different models to classify data – the first utilized only text as input, the second combined text with location and keyword, and the third used text and keyword without location. The model that processed only text inputs achieved the highest accuracy, with a public score of 0.77413 on Kaggle. In contrast, the model incorporating text, location, and keyword yielded a score of 0.76064, while adding only the keyword to the text resulted in a slightly improved score of 0.76218, still below the text-only model.

Kaggle Submission



My Kaggle username is riteshrk (link). The notebook has been uploaded on Kaggle and can be accessed here. The results of the submission were submitted on Kaggle and can be accessed here.

Code