Assessing Emotional Intensity in Tweets using Deep Learning

Abstract: This technical paper focuses on determining the strength of emotions expressed in tweets. We assign a score between 0 and 1 to measure the intensity of a specific emotion, where a score of 1 represents the highest intensity and 0 represents the absence of that emotion. Each tweet, combined with a particular emotion, is treated as a case for analysis. It's important to note that the scores are used comparatively rather than having a fixed meaning. Our approach involves preprocessing the tweets by cleaning and organizing the text, and then using advanced language processing techniques with the help of BERT models. We train a neural network with dense layers and dropout to predict the emotion intensity score. The paper includes experimental results, training processes, and discusses the potential applications of accurately measuring emotional intensity in tweets

Data: In this technical paper, we utilize training and test datasets to analyze four emotions: joy, sadness, fear, and anger. The training dataset consists of tweets accompanied by real-valued scores ranging from 0 to 1, indicating the level of emotion experienced by the speaker. For example, the anger training dataset contains tweets along with corresponding scores reflecting the intensity of anger. In contrast, the test data includes only the text of the tweets, without associated intensity scores. This data enables us to train our model and evaluate its ability to accurately predict the intensity of emotions based solely on the content of the tweet.

Data Loading and Preprocessing:

- The code starts by loading the dataset from a file using pandas and assigns appropriate column names.
- Various preprocessing steps are applied to the 'tweet' column using regular expressions and NLTK library functions. These steps include removing URLs, mentions, special characters, numbers, and converting emojis to text representation.
- Tokenization is performed on the 'tweet' column using the NLTK TweetTokenizer

Feature Extraction:

- The code then utilizes the BERT (Bidirectional Encoder Representations from (Transformers) model and tokenizer from the Hugging Face library.
- The 'features' column is created by passing each tokenized tweet through the BERT model to extract the last hidden state features.
- The resulting features are converted to NumPy arrays.

Padding and Reshaping:

In natural language processing tasks, it is essential to preprocess textual inputs through padding, truncation, and reshaping techniques. Padding sequences using the `pad_sequences` function from the `tensorflow.keras.preprocessing.sequence` module allows for equal sequence lengths by adding special tokens, typically zeros. This step ensures consistency in the input dimensions across all sequences.

Truncation, on the other hand, is employed when sequences exceed a predetermined maximum length. It involves discarding elements beyond the specified length, thereby enforcing uniformity among sequences. By applying padding and/or truncation, the sequence data is transformed into a structured 2D array format, where each row corresponds to a sequence. This reshaping process makes the data compatible with neural network models and optimizes their performance.

By incorporating these preprocessing steps, the model can effectively process textual inputs and yield accurate and reliable results in various natural language processing applications

Neural network and evaluation: The implemented model for this task utilizes the TensorFlow and Keras libraries. It is a sequential model consisting of multiple dense layers with different activation functions. The model architecture includesl2 regularization, dropout layers to prevent overfitting.

The input layer has a shape compatible with the data, and subsequent dense layers are added with progressively decreasing units. The final dense layer has a single unit with a linear activation function, suitable for regression tasks.

To optimize the model's performance, an early stopping callback is incorporated. It monitors the mean squared error and stops training if no improvement is observed within a specified patience value. The model is compiled using the Adam optimizer and mean squared error as the loss function and evaluation metric, respectively.

By employing this model architecture and optimization techniques, the model can effectively learn and make predictions on various regression tasks.

Model evaluation

JOY	Training	Validation/development
MSE	0.0192	0.0353
SADNESS	Training	Validation/development
MSE	.0325	.050
ANGER	Training	Validation/development
MSE	0.022	0.0272
FEAR	Training	Validation/development
MSE	0.0302	0.0330

Assessing Emotional Intensity in Tweets using statistical model

Abstract:

. In this technical paper, we explore two approaches for sentiment analysis of tweets: a lexicon-based approach and a Word2Vec-based approach. We leverage popular libraries and methods to implement these approaches and compare their performance.

Data Loading and Preprocessing:

- The code starts by loading the dataset from a file using pandas and assigns appropriate column names.
- Various preprocessing steps are applied to the 'tweet' column using regular expressions and NLTK library functions. These steps include removing URLs, mentions, special characters, numbers, and converting emojis to text representation.
- Tokenization is performed on the 'tweet' column using the NLTK TweetTokenizer

Feature Extraction:

- The code then utilizes the BERT (Bidirectional Encoder Representations from (Transformers) model and tokenizer from the Hugging Face library.
- The 'features' column is created by passing each tokenized tweet through the BERT model to extract the last hidden state features.
- The resulting features are converted to NumPy arrays.

Padding and Reshaping:

In natural language processing tasks, it is essential to preprocess textual inputs through padding, truncation, and reshaping techniques. Padding sequences using the `pad_sequences` function from the `tensorflow.keras.preprocessing.sequence` module allows for equal sequence lengths by adding special tokens, typically zeros. This step ensures consistency in the input dimensions across all sequences. Truncation, on the other hand, is employed when sequences exceed a predetermined maximum length.It involves discarding elements beyond the specified length, thereby enforcing uniformity among sequences. By applying padding and/or truncation, the sequence data is transformed into a structured 2D array format, where each row corresponds to a sequence. This reshaping process makes the data compatible with neural network models and optimizes their performance.

By incorporating these preprocessing steps, the model can effectively process textual inputs and yield accurate and reliable results in various natural language processing applications

Model

we use linear regression to model with l2 regularisation

Model evaluation

JOY	Testing	Validation / Development
MSE	0.023	0.035
MAE	0.123	0.16
SADNESS	Testing	Validation / Development

MSE	0.024	0.028
MAE	0.129	0.14
ANGER	Testing	Validation / Development
MSE	0.018	0.022
MAE	0.108	0.11
FEAR	Testing	Validation / Development
MSE	0.026	0.030
MAE	0.128	0.143