Emotion Classification from Tweets

By: Isabel Heard, Chiratidzo Sanyika, Tigist Tefera, Niru Shanbhag, and Carol Jiang

Introduction:

Our project is based on a data set from Kaggle, where we will be doing sentiment analysis on a collection of Twitter messages that show six different emotions. The use of this data set is to explore the nuanced emotional landscape within the world of social media. For our project, we focused on applying natural language processing techniques to determine which model performs the best based on its accuracy. The models that we used were: Logistic Regression, Naive Bayes, LSTM (Long Short-Term Memory), RNN, and CNN.

Dataset Description:

The data consists of three different variables: *ID*, *text*, and *label*. *ID* is a unique id number assigned to each row. The *text* variable is the Twitter message. Lastly, the *label* variable, which ranges from zero to 5. These consist of sadness (0), joy (1), love (2), anger (3), fear (4), and surprise (5). This dataset does not contain null values.

During the data cleaning process, several transformations were applied to enhance the quality and usability of the dataset. Initially, the data was then restructured to replace instances of '1' with 'Sadness' for clearer emotional labeling. Subsequently, URLs were stripped from the text to eliminate irrelevant web addresses. Extra white spaces were also removed to maintain consistent formatting. Special characters, punctuation, and abbreviations were eliminated to focus solely on meaningful text content. Additionally, all text was converted to lowercase to standardize the data. Common stop words were removed to reduce noise and emphasize important words. Lastly, emojis within the text were translated into their corresponding textual representations to integrate them seamlessly into the analysis.

In order to ensure that each emotion category has an equal representation in the dataset, We randomly select 14,000 rows from each emotion category, then split data into training (70%), development (15%), and test (15%).

Baseline Approach:

Our baseline approach for solving the sentiment analysis problem was to run different models with no hyperparameter adjustments. For our logistic regression model with no parameter adjusting, we achieved an accuracy score of 86%. For our Naive Bayes model, we achieved an accuracy score of 85%. With these baseline scores, we were ready to adjust our models and try some new ones to see if we could improve these scores.

Methods Description & Evaluation:

Logistic Regression A Model

We first ran a logistic regression on the dataset. We performed a grid search, using the sklearn GridSearchCV to see which hyperparameters would best fit our data. Those parameters ended up being: {'C': 10, 'penalty': l2, 'solver':lbfgs, 'max_iter': 1000, 'class_weight': balanced}. This information was then used to create a model to run sentiment analysis. We determined that the accuracy of this model on the development set was 87%.

Logistic Regression B Model

We then tried another logistic regression model, but changed our approach with the data cleaning aspect. This data set included the stop words, special characters and punctuation. Using the default hyperparameters for logistic regression, we got an accuracy of 91.26% on the development set. After tuning the model on the development set, our ideal parameters were: {'C': 1 'penalty': l1, 'solver': liblinear, 'max_iter': 1000, 'class_weight': None}. This brought the accuracy on the development set up to 92.06%. Using this new classifier, we achieved a test accuracy of 92.48%. The model is most confident with its classification of tweets that convey surprise. The model is least confident with its classification of tweets that convert Joy.

Naive Bayes Model

Next we ran a Naive Bayes model to see if that would fit our data well. We once again ran a grid search to find our parameters. These came out to be: {alpha: 0.5, fit_prior: False}. With this, our model only achieved an accuracy score of 85% on the development set, which was no improvement from our baseline approach. So, Naive Bayes may not be the best model for our dataset.

LSTM Model

The model used an embedding layer as a first layer of the model to convert the input sequence into dense vectors of fixed size to represent words. Adds LSTM layer with 128 units to allow the model to learn more complex patterns and dependencies in the dataset. We used 'dropout = 0.2' and 'recurrent dropout = 0.2' parameters for regularization techniques to prevent overfitting. Adds Dense layer with a Softmax activation function for multi class classification. We set up the model for training using categorical cross-entropy as the loss function, the Adam optimizer for updating weights, and accuracy as the metric to monitor during training.

For training we set up early stopping to monitor the validation loss and restore the best weights when the loss stops improving after a certain number of epochs. Fits the model to the training data (X_train and y_train) for 20 epochs with a batch size of 128. The validation data (X_dev and y_dev) is used to monitor the performance of the model during training.

The performance of LSTM with the training and test data are:

Training => Accuracy: 0.9409 - Loss: 0.1516

Test => Accuracy: 0.9378 - Loss: 0.1575

RNN Model

The model first tokenizes the reviews to only include the top 1000 most frequent words. Then, to make the sequences the same length, the tokenized vectors were padded to the max length of the reviews. Next, the data was split into 70% training, 15% development, and 15% testing. The RNN model contains an embedding layer that takes 3 parameters: vocabulary length, embedding length, and input length. The vocabulary length specifies the number of unique words in the dataset. The input length specifies the size of the padded input sequence. The embedding length specifies the size of the output vector. Next, the Simple RNN is specified with a batch size of 128 with dropout = 0.2 and recurrent dropout = 0.2. The last layer in the RNN is a dense layer of 6 neurons, where the activation is set to softmax, since we have 6 different sentiments.

The RNN model was compiled with sparse categorical cross entropy, with the optimizer set to adam. When fitting the model, batch size of 128 and 20 epochs were used. During the training process, development sets were used to monitor accuracy improvement over epochs. Additionally, earlystopping was implemented to avoid overfitting the model on the training.

More specifically, validation loss was monitored, patience was set to 3, and the min delta was set to 0.001. This means that after 3 epochs, if the validation loss does not decrease by 0.001, the model will stop.

Training: accuracy: 0.9475 - val_loss: 0.2361 - val_accuracy: 0.9202

Testing: accuracy: 0.9094 loss: 0.3023

CNN Model

The model contains an embedding layer that utilizes dense vector embeddings to represent words and phrases. The core of the model architecture consists of convolutional layers with 64 filters and a kernel size of 3, which helps in extracting important features from the text data. ReLU activation functions are applied to these convolutional layers to introduce non-linearity. Batch normalization layers are included after each convolutional layer to aid in reducing internal covariate shift and accelerate the training process by normalizing the activations. Global max pooling layers are utilized to condense the output from the convolutional layers into a fixed-size representation, focusing on the most salient features across each feature map.

During training, a batch size of 256 is used over 25 epochs to optimize the model parameters. The reported performance metrics on both the training and testing datasets are as follows:

Training dataset:

Loss: 0.11 Accuracy: 0.96 Precision: 0.96 Recall: 0.96

Testing dataset:

Loss: 0.34 Accuracy: 0.9 Precision: 0.91 Recall: 0.9

Discussion:

Our comprehensive evaluation and comparison of Naive Bayes, Logistic Regression, RNN, CNN, and LSTM models for multi-class sentiment classification lead us to select LSTM as the best model. Its superior performance, robustness to imbalanced data, and ability to handle complex textual patterns make it the most suitable choice for accurately classifying sentiment across multiple classes.

We also looked at examples where our baseline model was confident and uncertain. We found that examples with high accuracy included key words, such as ‘humiliated’ being associated with surprise, ‘threatened’ being associated with fear, or ‘lucky’ being associated with love. Our model was more uncertain when there were certain words that could relate to one or more emotions. Such as the tweet, ‘feel excited’, our model thought that was joy, but in fact it was classified as surprise. Another example of an uncertain classification was the tweet, ‘feel like stupid immature people in high school could grow old together.’ Our model thought it was conveying sadness, yet it was classified as anger. This goes to show it can be difficult to do sentiment analysis on tweets, especially when looking at six different emotions, there is a lot of room for crossover.

Conclusion:

In this project we were planning to build a model capable of accurately predicting the sentiment expressed on the text across multiple emotion categories. Through experimenting and analyzing the performance of models of Naive Bayes, Logistic regression A, Logistic regression B, LSTM, RNN and CNN, we identified the effectiveness of LSTM models in capturing complex patterns and dependencies in sequential text data.

Our findings indicate that the LSTM model achieved commendable performance on the sentiment classification task, with an overall accuracy of 93.78% on the testing dataset. We also observe that certain emotion categories such as ‘Love’ exhibited highest accuracy compared with others:

Accuracy for emotion 0(Sadness): 0.94 (2109 instances)

Accuracy for emotion 1(Joy): 0.90 (2030 instances)

Accuracy for emotion 2(Love): 0.98 (2161 instances)

Accuracy for emotion 3(Anger): 0.96 (2147 instances)

Accuracy for emotion 4(Fear): 0.87 (2064 instances)

Accuracy for emotion 5(Surprise): 0.97 (2089 instances)

This indicates that distribution of sentiment expression and unique linguistic features helps to distinguish one emotion from the others.

In Conclusion , from this analysis we gained that various applications like social media analysis, customer feedback analysis, and market sentiment analysis require accurate sentiment classification for understanding human emotions and behaviors in text data.

Appendix:

Isabel Heard: Code, Paper & Presentation

Chiratidzo Sanyika: Code, Paper & Presentation

Tigist Tefera: Code, Paper & Presentation

Niru Shanbhag: Code, Paper & Presentation

Carol Jiang: Code, Paper & Presentation