Emotion Detection from Text: Exploring Machine Learning, Deep Learning, and Transformer Approaches

Northeastern University
CS6120 - Natural Language Processing
Group 2 - Ngoc Khanh Vy Le
12/8/2024
Github

Abstract

Emotion detection from text is a critical task in natural language that has a lot of applications in different fields ranging from sentiment analysis, monitoring mental health, customer feedback or building chatbots. There are different algorithms, different techniques and datasets to build the emotion detection model. In this report, I use the dataset from GoEmotions to analysis and compare the performances of three approaches: Logistic Regression served as a baseline model, Long Short-Term Memory (LSTM) as a deep learning approach, and Bidirectional Encoder Representations from Transformers (BERT) as a transformer-based approach. I use the labeled dataset GoEmotions from Google to train and evaluate all the models using accuracy, F1-score, precision and recall. The results demonstrate that Logistic Regression provides a strong baseline at 63%. Since the LSTM models have slightly higher performance at 63%, the BERT model performs slightly better than both, at 65% of accuracy.

1. Introduction

These days people prefer communicating through text messages rather than in-person. However, the lack of non-verbal cues such as tone of voice, facial expressions, and body language makes it challenging to accurately interpret the emotions conveyed through text messages. Detecting emotion from text is essential research in solving this problem. By applying machine learning and natural language processing techniques, we can detect and analyze the sender's feelings through their text messages. In addition, identifying emotions like happiness, sadness, anger, or fear, these systems can enhance human-computer interactions, provide better insights into user behavior, and support decision-making in a variety of applications.

Over the past few years, there have been various emotion detection models developed using different techniques. The main objective of this paper is to conduct a comparative analysis of three popular models—Logistic Regression, LSTM, and BERT—for emotion detection from text using the same dataset with the same data preprocessing process. By evaluating their performance on a standard dataset, my goal is to find which models perform better in terms of accuracy and other metrics and build the predictive emotion model based on the best model which allows the users to check the emotions based on the input texts.

This report focuses on models trained and evaluated on publicly available datasets relevant to emotion detectionI. I use the robust dataset GoEmotions from researchers from Google Research. I choose the dataset because of its comprehensiveness, balance, and availability. For the comparisons between models, I compare their training times, interpretability, and overall performance to understand each model's strengths and limitations and then choose the best among these three to create a robust emotion detection model.

2. Dataset

2.1 GoEmotions

The dataset is created by Google Research. The data can be downloaded from Github. It has 58000 corpus which are carefully extracted from comments in Reddit. The dataset has 27 emotion labels and neutral labels. All labels are grouped into 6 main universal emotions and neutral emotions. Each corpus has more than one label. The dataset is clean, well-documented and has been divided into 3 separate files: training, testing and validation dataset.

2.2 Glove

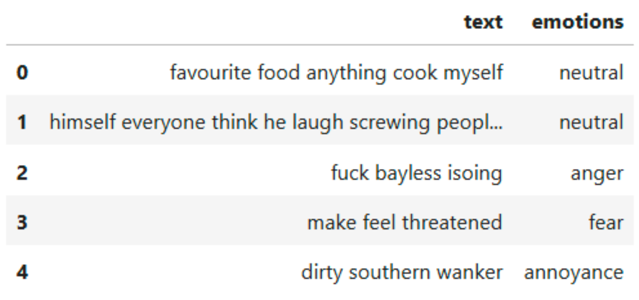
I use *glove.6B.100*d dataset from Glove to use in the LSTM model to convert words into dense vector representations. The dataset I used in this report is from Kaggle.

3. Methodology

The Logistic Regression model serves as a baseline, while the primary focus is on implementing and comparing various configurations of LSTM models with different numbers of layers and sequence lengths. Each model is evaluated based on accuracy, precision, recall, and F1-score to assess their performance comprehensively.

3.1 Data Preprocessing:

The initial dataset has three columns: corpus, emotion id and the corpus id. First step, I created two different columns which contain the list of emotion ids and number of emotions of each corpus. I used the emotions file from the data folder to convert the emotion ids into the emotion text for all three datasets.



	text	emotions
0	favourite food anything cook myself	neutral
1	himself everyone think he laugh screwing peopl...	neutral
2	fuck bayless isoing	anger
3	make feel threatened	fear
4	dirty southern wanker	annoyance

Table 1: Example of the final preprocessed dataset

I cleaned the dataset by removing stopwords, numbers, punctuation, URLs, emojis, and short corpora, converting all words to lowercase, and then applying lemmatization to all three datasets. Drop the neutral, admiration, approval, gratitude, amusement. Map the emotion label using *ekman_mapping* file from the data folder which reduces the label size from 28 to 6. I then expanded the dataset: for corpus that evaluated with two or more labels will be expanded into different rows with the same corpus text but different labels and convert the value of emotions columns from type of list to just string. Drop all na values, and drop text values that are not string type. Remove duplicates and unnecessary columns. The final data will have just the corpus and its own emotion label.

3.2 Feature Engineering

a. **TF-IDF:** Textual data was transformed into numerical representations using TF-IDF. Key configurations included: a maximum of 10,000 features to limit dimensionality and Bi-gram range (ngram_range=(1, 2)) to capture context from adjacent words.

b. **Label Encoding and One-Hot Encoding:** Emotion labels were encoded from text to int using LabelEncoder and transformed into one-hot encoded vectors for multi-class classification.

c. **Embedding Matrix:** I used pre-trained GloVe embeddings (*glove.6B.100d.txt*) to initialize an embedding matrix, mapping the vocabulary to a 100-dimensional vector space.

d. **Tokenization:** After calculating the maximum length was calculated, text data was tokenized using the Tokenizer class with the oov_token parameter set *UNK* to convert raw text data into numerical representations, then sequences were padded to a maximum length of 145 words for uniformity.

e. **Early Stopping:** I implemented the early stopping after 4 interactions and calculated based on the validation loss rate. This will improve training efficiency and prevent overfitting.

f. **Padding:** LSTM models and Transformers like DistilBERT or BERT require input sequences of a fixed length for batch processing. I applied padding to ensure that all input sequences in a batch are of the same length by appending special tokens (usually zeros) to shorter sequences or truncating longer ones.

3.3 Model Implementation

3.3.1 Logistic Regression

The model was selected due to its simplicity and effectiveness for text classification tasks. I used max_iter=1000 to ensure convergence during optimization

3.3.2 LSTMs

I implemented and fine-tuned multiple variants of LSTM models, each employing unique configurations to identify the optimal approach. All models utilized pre-trained GloVe embeddings as a non-trainable embedding layer, providing rich, contextualized representations for input text.

i. **Bidirectional LSTM**: This model utilized a stack of three Bidirectional LSTM layers to capture sequential dependencies and context in both forward and backward directions. A fully connected dense layer with softmax activation produced the final sentiment classification.

ii. **Bidirectional LSTM with Regularization**: In this mode, I added L2 regularization to the dense output layer and increased dropout rates to enhance generalization. In addition to lower learning rate (0.001) which is to ensure stable training. This model aimed to assess the impact of regularization on performance

iii. **Enhanced LSTM with Additional Dense Layer:** I added an additional dense layer with ReLU activation before the final output layer to improve feature extraction.I choose a fine-tuned using a lower learning rate of 0.0001 and gradient clipping to handle potential vanishing/exploding gradients in deep LSTM layers

iv. **Convolutional LSTM:** I used a hybrid approach for this model. I combined a Conv1D layer for local feature extraction with LSTM layers to incorporate temporal dependencies. For pooling, I used global max pooling to reduce the dimensionality before fully connected layers and I increased dropout to mitigate overfitting in the dense layers

v. **Multi-Head Attention LSTM:** I used Multi-Head Attention layers to capture contextual relationships between words more effectively. In addition, I also applied global average pooling to summarize sequence features and used pertain glove embedding dataset to avoid overfitting

3.3.2 Fine-tuned BERT

My third approach is implementing a BERT-based sentiment classification model by fine-tuning a pre-trained DistilBERT model. The goal was to leverage the rich, contextualized word representations from the pre-trained model and adapt it to the specific task of sentiment analysis.

To optimize performance, I implemented several fine-tuning approaches. The model involved training the entire DistilBERT model with a linear classifier on top using the AdamW optimizer and a learning rate of 0.005. The classification head comprised a fully connected dense layer with a softmax activation function to predict the sentiment classes. This base model utilized DistilBERT's contextual embeddings to capture semantic nuances in the text data.

Additionally ,I used regularization techniques, including dropout layers with a rate of 0.3 and L2 regularization, to enhance generalization and mitigate overfitting. Also, I utilized a cosine annealing learning rate scheduler to ensure smooth convergence during training. The text data was tokenized using DistilBERT's tokenizer, with padding and truncation to a fixed sequence length of 64 for consistent batch processing.

3.4 Model Evaluation

The trained models were evaluated on the validation dataset using a variety of metrics, including accuracy, precision, recall, and F1-score, to comprehensively compare their performance and identify the best-performing model. Furthermore, I used the test dataset to evaluate its generalization capability, using similar metrics.

k-fold. k-fold cross-validation (with k = 5) was performed on the training data to check a robust assessment of the models' predictive capabilities across different data splits

Loss function. Categorical cross-entropy was utilized as the loss function for all models, appropriate for multi-class classification.

4. Results and Discussions

4.1 Exploratory Data Analysis

Figure 1 shows that the *neutral* emotion is by far the most frequent emotion in the dataset, significantly outnumbering all other emotions. This suggests an imbalance in the dataset, which could affect the performance of models trained on this data. Emotion labels like pride relief and grief occur rarely in the dataset, indicating underrepresentation of these categories.

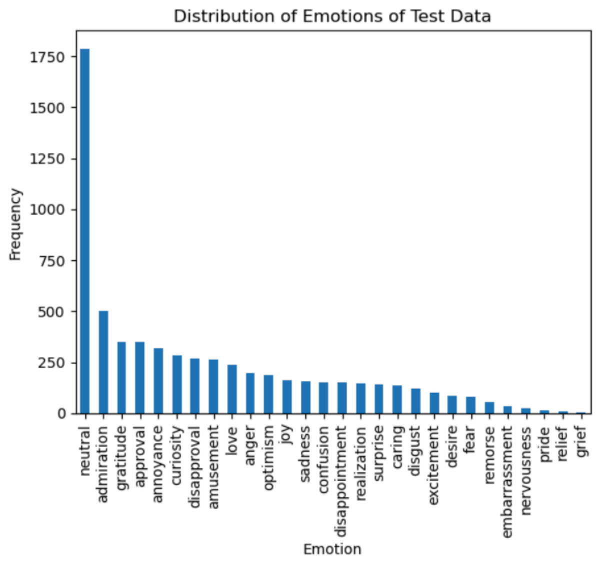


Figure 1: The distribution of emotion labels

After cleansing and applying ekman-mapping, the sentiment *joy* is the most frequent category, showing a substantial representation compared to other sentiments suggesting that the training data is heavily skewed toward positive emotions. As expected, *disgust* and *fear* are the least frequent sentiments, indicating a significant underrepresentation (*Figure 2)*

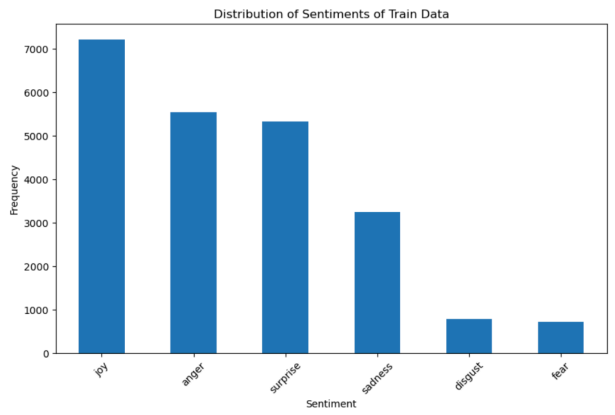


Figure 2: The distribution after ekman-mapping

4.2 Logistic Regression

The logistic regression model has a validation accuracy and testing accuracy, around 63%, which indicates the model performs reasonably well, as the validation and test accuracies are almost identical. The *joy* label has the highest performance with an F1-score of 74% on both validation and on test, likely because it is the most frequent class. However, the model is hard to classify the minority classes like *disgust* and *fear*.

Validation Accuracy	62.938%
Testing Accuracy	63.064%
Mean Accuracy	0.6157 ± 0.0025

Table 2: cross-validation scores of Logistic Regression

I applied 5-k-fold cross validation to check the robustness of the model. The small variance indicates stable and consistent performance across folds, suggesting that the model is not overfitting. As the Logistic Regression is a baseline model and it performs well, I expect deep learning models will also perform relatively well.

4.3 LSTMs

Emotions	**Precision**	**Recall**	**F1**
anger	0.61	0.57	0.59
disgust	0.59	0.46	0.52
fear	0.63	0.58	0.61
joy	0.71	0.79	0.74
sadness	0.58	0.55	0.57
surprise	0.57	0.59	0.58
accuracy			**0.63**
macro avg	**0.62**	**0.59**	**0.60**
weighted avg	**0.63**	**0.63**	**0.63**

Table 3: Precision, recall and F1-score of each label of Bidirectional LSTM model

I ran five different LSTM models with different optimization, different layers and learning rate to find the best LSTM model. My best performing model is the Bidirectional LSTM model with the accuracy of 63%, which is the same Logistic Regression model performance. Meanwhile, the convolutional model is the weakest performer across all metrics, accuracy at 57%, suggesting that CNNs may not effectively capture temporal dependencies in the dataset compared to other LSTM models.

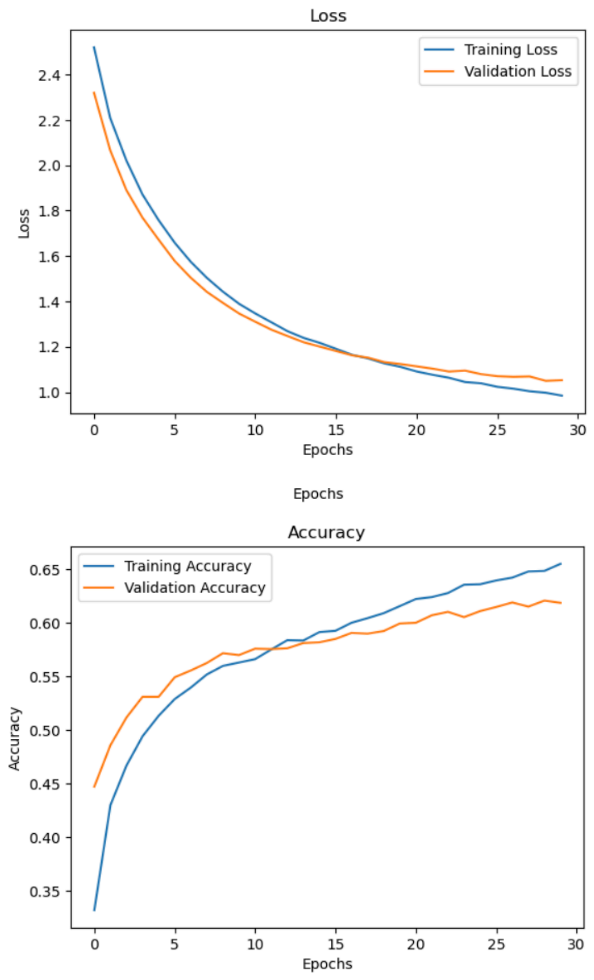


Figure 3: The loss and accuracy of Bidirectional LSTM

Table 3 shows the model accuracy is 63% which is reasonably well. However, the model does not significantly outperform a simpler Logistic Regression model, indicating possible limitations in the dataset or feature representation. Across the performance across emotion labels, *joy* has the best performance among other labels with the highest precision 71%, recall 79% and F1-score 74%, likely due to clear feature patterns and sufficient data representation for this class. In addition, two graphs showing *Figure 3* demonstrate that there is no overfitting in the model.

The 5-fold cross-validation results for the Bidirectional LSTM model in *Table 4* shows the consistent performance across the folds. All the average scores - precision, recall and f1 are similar which indicate that the model's predictions are both balanced and reliable.

Average Accuracy	0.6972272125174932
Average Precision	0.6947191416737543
Average Recall	0.6972272125174932
Average F1-Score	0.694865203573855

Table 4: 5-fold cross-validation results for the Bidirectional LSTM model

4.3 BERT

My BERT model has an accuracy of 66% which is lower than expected but the model still performs better than the other two approaches. Three epochs are run in this model. The loss results of each epoch are 0.8186, 0.4967, and 0.3472 respectively. The significant reduction in loss over epochs shows the model is learning effectively and converging well during training. This shows BERT's ability to leverage its transformer architecture to capture contextual and sequential dependencies in text effectively.

Emotions	**Precision**	**Recall**	**F1**
anger	0.64	0.61	0.62
disgust	0.57	0.42	0.52
fear	0.64	0.74	0.69
joy	0.78	0.77	0.76
sadness	0.55	0.60	0.57
surprise	0.62	0.65	0.63
accuracy			**0.66**
macro avg	**0.63**	**0.54**	**0.63**
weighted avg	**0.66**	**0.66**	**0.66**

Table 5: Precision, recall and F1-score of each label of fine-tuned BERT model

Similar to other models, *joy* label performs the best while *sadness* has the worst performance (*Table 5*).. In addition, The confusion matrix in *Figure 4* reveals notable misclassifications, such as *anger* often being mistaken for surprise and joy for surprise. This result suggests that there are overlapping features in these emotions.

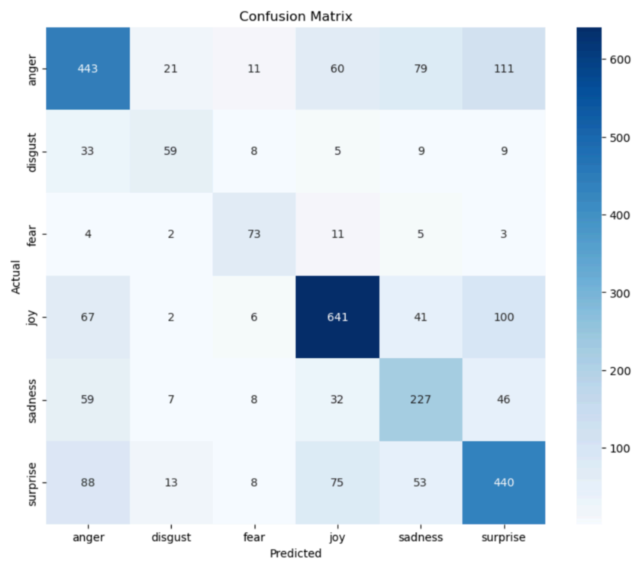


Figure 4: Confusion matrix of BERT model

5. Conclusion

Overall, based on *Figure 1* and *Figure 2*, we can see that the dataset is extremely imbalanced and right skewed - neutral and joy labels have the highest occurrences. This imbalance likely effects model performance, making it harder for models to accurately classify minority emotions. These imbalances also skew evaluation metrics like accuracy, favoring predictions aligned with majority classes.

Although LSTM models demonstrate the ability to capture temporal and sequential dependencies better, their advantages are constrained by the dataset's limitations. Furthermore, while BERT’s transformer-based architecture excels at capturing contextual relationships in text, its performance is constrained by dataset limitations, particularly class imbalance. The model struggles with underrepresented classes, which hinders its ability to generalize effectively to minority emotions

6. Future Improvement

The results from the three models show similar performance. This performance parity suggests that the characteristics of the dataset, such as class imbalance or limited distinguishing features, may be the primary constraints rather than the choice of model. The slight improvement observed with LSTM and BERT highlights their potential to capture sequential and contextual relationships, but their lack of significant advantage suggests that the dataset size or diversity may be insufficient for these advanced models to fully leverage their capability. Creating a larger, more diverse datasets could help address the limitations of the current dataset.

In addition, to improve performance, we should address class imbalance through techniques like oversampling or class weighting and the dataset to include more examples of underrepresented emotions is crucial. Furthermore, fine-tuning BERT with optimized hyperparameters and leveraging ensemble methods could help push performance beyond the current plateau. Besides metrics used in this report, we should also use different metrics to better evaluate performance on imbalanced classes. We can also consider different techniques such as transfer learning from related tasks, hierarchical classification for overlapping emotions, or adopting few-shot learning approaches for minority classes could further improve performance.

References

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. GoEmotions: A Dataset of Fine-Grained Emotions. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)

Mohamed Zul Fadhli Khairuddin, Suresh Sankaranarayanan, Khairunnisa Hasikin, Nasrul Anuar Abd Razak, Rosidah Omar, Contextualizing injury severity from occupational accident reports using an optimized deep learning prediction model, PeerJ Computer Science, 10.7717/peerj-cs.1985, 10, (e1985), (2024)

Nandwani, P., Verma, R. A review on sentiment analysis and emotion detection from text. *Soc. Netw. Anal. Min.* **11**, 81 (2021). https://doi.org/10.1007/s13278-021-00776-6

Machová K, Szabóova M, Paralič J and Mičko J (2023) Detection of emotion by text analysis using machine learning. Front. Psychol. 14:1190326. doi: 10.3389/fpsyg.2023.1190326

Haryadi, Daniel & Putra, Gede. (2019). Emotion Detection in Text using Nested Long Short-Term Memory. International Journal of Advanced Computer Science and Applications. 10. 10.14569/IJACSA.2019.0100645