SentiMind: A Deep Neural Network Based Emotion Classifier

Zixi Qu, Jinyan Yi, Kangzhi Gao
The Edward S. Rogers Sr. Department of
Electrical Computer Engineering
University of Toronto
{zixi.qu, alexjy.yi, kangzhi.gao}@mail.utoronto.ca

Abstract

SentiMind is a emotion classifier, which leverage the power of deep neuron network, to generate a sentiment analysis of sentences. This model can be practically helpful to summarize social media comments, analyzing public opinion trends, identifying conflicts and strengthening the quality of verbal communication. SentiMind will involve multiple deep learning techniques and incorporate with online sentiment dataset from Hugging Face to produce reliable and rapid sentiment analysis to a string of words. (String of words: one or more sentences).

Assentation of Teamwork

- **Zixi:**
  - Construct the main architecture of the project.
  - Implement and conduct performance experiments with CNN architecture.
  - conduct performance experiments of upstream encoder transformer.
- **Jinyan:**
  - Implemented and Fine-Tuned LSTM-based Model
  - Worked with Zixi in writing poster and presentation.
- **Kangzhi:**
  - Investigated proper methods to calculate accuracy.
  - Implemented the infer and accuracy calculation functionality.
  - Implement final report.

1 Introduction

It is crucial to understand the emotions behind human-written text in this modern era, as what we express online reflects our knowledge, positions, and sentiments. These sentiments can quickly spread, influencing decisions and mindsets. Our project, SentiMind, is a deep learning project that aims to address the sentiment behind every human text, and accurately classify the true tone of the input. We have two main approaches for our implementation: one uses a convolutional neural network (CNN) and the other uses a Long Short-Term Memory (LSTM) recurrent neural network. Both models begin by tokenizing input sentences and applying a transformer encoder to produce context-aware embeddings. The models will be trained on Huggingface labelled datasets to ensure accuracy. We expect SentiMind to offer a convenient and efficient way to detect the sentiment behind every human writing and provide a practical solution to understanding emotional nuances in text.

ECE 1508: Applied Deep Learning (Fall 2024).

2 Preliminaries and Problem Formulation

2.1 Objective

The objective of this project is to develop a machine learning model that accurately identifies and categorizes emotions expressed in text. The model will take a string of words as input and output the corresponding emotion category.

- **Input:** A string of text.
- **Output:** One or more emotion category ID, as detailed in Appendix A which maps each ID to a specific emotion such as Disappointment, Admiration, Amusement, or Joy.
- **Example:**
  - **Input:** "Lol so petty, I kinda love it. I probably wouldn't actually do that but it's tempting."
  - **Output:** 0 (Admiration), 1 (Amusement)
  - **Input:** "No problem at all, glad I could help :) Cheers!"
  - **Output:** 17 (Joy)

2.2 Transformer Encoder

As mentioned in the introduction, our models use a transformer encoder. It takes tokenized text as input and outputs context-aware embeddings, utilizing a self-attention mechanism to capture relationships between words in the entire sequence. This results in embeddings rich in contextual information, essential for accurately identifying nuanced emotional expressions.

2.3 Accuracy Metric: Jaccard Index

In our project, we used the Jaccard Index as the accuracy calculation metric, which is defined in the following:

$$J(\text{Label}, \text{Prediction}) = \frac{|\text{Label} \cap \text{Prediction}|}{|\text{Label} \cup \text{Prediction}|}$$

This formula is particularly well-suited for our multi-label sentiment analysis task. The numerator, $|\text{Label} \cap \text{Prediction}|$, represents the intersection of the true labels and the labels predicted by the model, indicating the number of correct predictions. The denominator, $|\text{Label} \cup \text{Prediction}|$, is the union of the true labels and the predicted labels, which includes all unique labels identified by both the ground truth and the predictions. This encompasses correctly predicted labels, false positives (incorrect labels predicted by the model), and false negatives (true labels that the model failed to predict). By calculating the ratio of the intersection to the union, the Jaccard Index not only measures the proportion of correct predictions but also penalizes the model for any incorrect predictions, providing a thorough and nuanced evaluation of the model's ability to accurately and reliably capture emotional expressions in text.

Examples:

- Prediction: $\{19, 21\}$, Label: $\{21\}$
$$J = \frac{|\{19, 21\} \cap \{21\}|}{|\{19, 21\} \cup \{21\}|} = \frac{1}{2}$$
- Prediction: $\{19, 21\}$, Label: $\{21, 23\}$
$$J = \frac{|\{19, 21\} \cap \{21, 23\}|}{|\{19, 21\} \cup \{21, 23\}|} = \frac{1}{3}$$
- Prediction: $\{19, 21\}$, Label: $\{210, 230\}$
$$J = \frac{|\{19, 21\} \cap \{210, 230\}|}{|\{19, 21\} \cup \{210, 230\}|} = 0$$

3 Solution via Deep Learning

To develop SentiMind, we first split the dataset into training, validation, and test sets in proportions of 60%, 20%, and 20%, respectively. We utilize two approaches for modeling the emotional tone conveyed in text: Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks, both enhanced with transformer-based models. For text data preprocessing, we use a transformer encoder, selecting from options like "roberta-base", "albert-base-v2", "distilroberta-base", or "huawei-noah/TinyBERT_General_4L_312D". These models are our choices for generating word embeddings that effectively capture word meanings and semantic relationships. The final choice among these transformer encoders is based on their performance, which will be illustrated in the Numerical Experiments section. This approach ensures that we select the most effective model for our sentiment analysis needs.

After processing the text with either a CNN or LSTM, a Feedforward Neural Network (FNN) takes over to output final predictions. Each prediction comprises 28 probabilities, corresponding to 28 different emotional categories, which are derived by applying a sigmoid function to each output. We use binary cross-entropy as the loss function for this binary classification task, where the goal is to determine the presence of each emotional category in the sentences. For the actual classification of emotions, only the indices with probabilities greater than 0.5 are selected. These indices, ranging from 0 to 27, represent specific emotions as detailed in Appendix A.

For evaluating model performance, we use the Jaccard Index as the accuracy metric. This will be further explained in the Implementation section, where we discuss how this metric helps assess the overlap between the predicted emotional categories and the actual labels, providing a clear measure of accuracy and reliability. The training process involves feeding batches of these embeddings into the models, adjusting batch sizes and other hyperparameters during the validation phase to optimize performance and prevent overfitting. This ensures that the models generalize well to new, unseen data. Finally, we test the models on the test dataset to evaluate their accuracy, reliability, and generalization capabilities.

4 Implementation

4.1 1-layer CNN

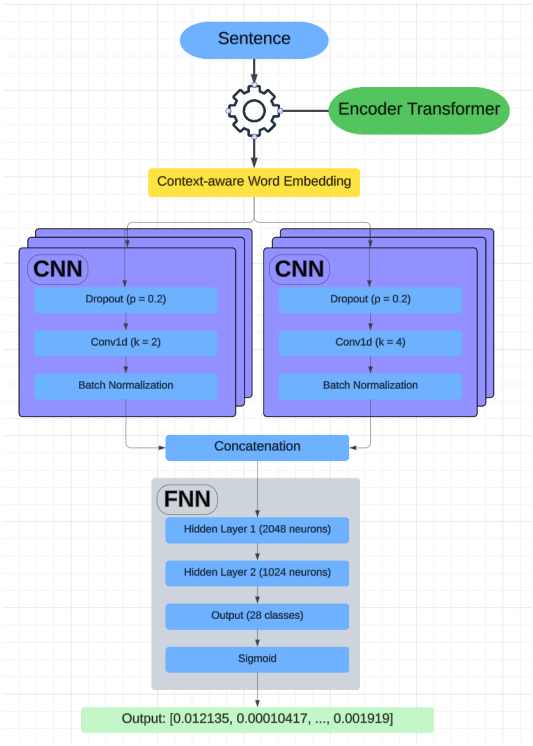

Figure 1: 1-layer CNN

For CNN implementation, we have developed both 1-layer and 2-layer CNN configurations. The Figure 1 illustrates the architecture for the 1-layer CNN, which begins with a sentence input that is processed through a transformer encoder, selected from models like "roberta-base", "albert-base-v2", "distilroberta-base", or "huawei-noah/TinyBERT_General_4L_312D". These encoders are important for generating context-aware word embeddings that accurately capture the semantics of the text.

In this 1-layer CNN architecture, the embeddings are then fed into two parallel convolutional paths. Each path incorporates a dropout layer with a rate of 0.2 to mitigate overfitting, followed by a convolutional layer with kernels set to sizes (2, embedding size) and (4, embedding size), respectively, and an output channel count of 256. These kernel sizes indicate that the CNN convolves every two and four word embeddings, capturing patterns over varying spans of text, with the stride set to 1. This setup is aimed at extracting a diverse range of features from the embeddings, and each convolutional layer is supplemented by batch normalization to enhance the model's training stability and efficiency.

Before concatenation, the output from each CNN path undergoes a max pooling operation, which helps in reducing the dimensionality of the feature maps and capturing the most significant features. After max pooling, the outputs from the CNN paths are concatenated. Before the concatenated feature map reaches the first hidden layer of the Feedforward Neural Network (FNN), a dropout layer with a dropout rate of 0.2 is applied to further prevent overfitting. The first hidden layer, consisting of 2048 neurons, processes these inputs and is followed by a ReLU activation function, which introduces non-linearity to the model. This is followed by batch normalization to stabilize the learning process, and another dropout layer to reduce overfitting.

The output of the first hidden layer feeds into the second hidden layer, also consisting of 1024 neurons. After processing through ReLU and batch normalization, the data passes to the output layer. This layer computes the raw predictions for 28 emotional categories, with each output transformed by a sigmoid activation function to convert these predictions into probabilities that indicate the presence of each emotional category in the input sentence.

4.2 1-layer LSTM

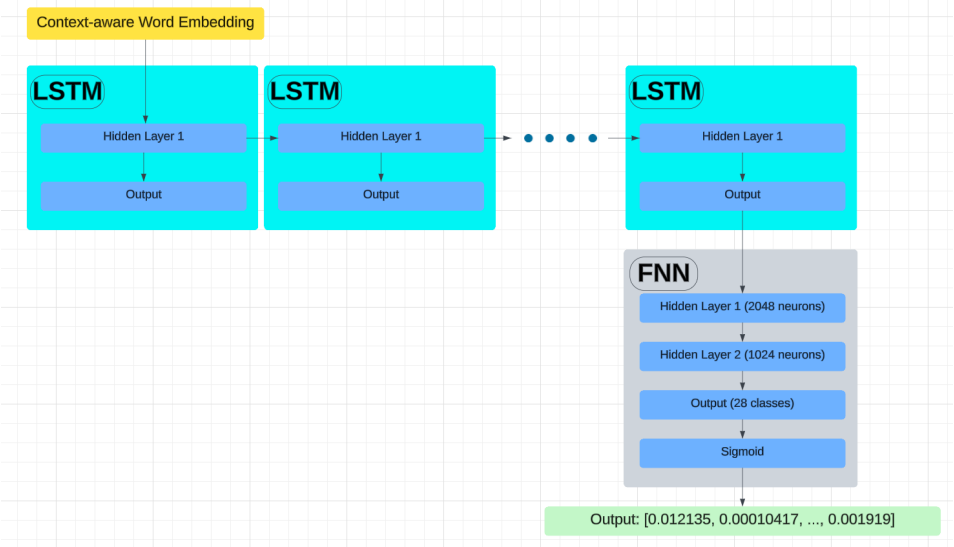

Figure 2: 1-layer LSTM

The architecture depicted in Figure 2 represents our 1-layer LSTM implementation. The sequence length in this setup corresponds to the number of words in the input sentence, aligning with the natural flow and structure of textual input.

The model begins with a single-layer LSTM that processes the context-aware word embeddings derived from the input sentences. As a single-layer network, no dropout is used in the LSTM layer. Following the LSTM, the model transitions to a series of fully connected layers. The first hidden layer processes the output from the LSTM, expanding it to 2048 neurons. This expansion is accompanied by ReLU activation to introduce non-linearity, followed by batch normalization and dropout to prevent overfitting and enhance generalization. The output from this first hidden layer then feeds into a second fully connected layer, which narrows the features down to 1024 neurons. This layer similarly uses ReLU activation, batch normalization, and dropout to maintain consistency in processing and to ensure the model's robustness and stability against overfitting.

The process culminates with the final output layer, which maps the 1024 neurons to 28 distinct classes corresponding to different emotional categories. A sigmoid activation function is applied at this stage, converting the logits into probabilities. This transformation is essential for the multi-label classification task, enabling the model to output a probability for each emotional category, indicating the presence or intensity of each emotion in the input text.

4.3 2-layer CNN

We have also implemented a 2-layer CNN which has a structure very similar to the 1-layer CNN, with a key difference in the replication of the CNN components. In the 2-layer CNN, each of the two paths of the initial CNN component is followed by an additional layer of CNN processing. This second layer replicates the structure of the first, including the dropout settings, convolutional layers, and batch normalization.

This means that after the first set of convolutional operations on the context-aware word embeddings, each path undergoes another sequence of convolutional processing before reaching the max pooling stage.

After the second convolutional layer in each path, the outputs proceed to the max pooling step as in the 1-layer CNN. The pooled features from both paths are then concatenated. This concatenated output feeds into the subsequent layers of the network, following the same flow through dropout, the Feedforward Neural Network (FNN), and finally to the output layer where the emotional probabilities are computed using a sigmoid function.

4.4 3-layer LSTM

We have also implemented a 3-layer LSTM, which maintains a very similar structure to the 1-layer LSTM but with additional complexity to enhance its learning capacity. The primary difference lies in the number of LSTM layers; the 3-layer LSTM features three sequential LSTM layers instead of one.

Just like the 1-layer model, the 3-layer LSTM processes the context-aware word embeddings derived from the input sentences. However, in the 3-layer setup, each LSTM layer is stacked on top of the previous one. This stacking enhances the model's ability to interpret emotional tones from text.

Following the LSTM layers, the model uses the same series of fully connected layers as the 1-layer LSTM. The output from the last LSTM layer passes through these dense layers, which include settings for ReLU activation, batch normalization, and dropout, just as in the 1-layer setup.

The final output layer maps the features to 28 distinct classes, each representing a different emotional category, with a sigmoid activation function applied to convert the logits into probabilities. This indicates the presence or intensity of each emotion in the input text.

5 Numerical Experiments

Model	Top Accuracy	Training Speed
roberta-base	38%	8it/s
albert-base-v2	40%	4it/s
distilroberta-base	24%	10.75it/s
huawei-noah/TinyBERT	38%	41.80it/s

Table 1: Encoder Model Performance

As mentioned earlier, the encoder transformer in our models has multiple choices as listed in Table 1. Considering both accuracy and training speed, 'Albert-base-v2' stands out with the highest accuracy of 40% and a moderate training speed of 4it/s. Although 'roberta-base' and 'huawei-noah/TinyBERT' also achieve an accuracy of 38%, with 'huawei-noah/TinyBERT' featuring a significantly faster training speed at 41.80it/s, training speed is less important in our scenario. We prioritize top accuracy to ensure the most reliable sentiment analysis. 'Distilroberta-base', despite its faster training speed of 10.75it/s, has a lower accuracy of 24% and thus does not meet our requirements. Given our focus on achieving the highest accuracy, we have chosen 'albert-base-v2' as our final choice for the transformer encoder in our project, as it provides the best performance according to our priorities.

Model	Train Accuracy	Test Accuracy
1-layer CNN	**87.72%**	**40.38%**
2-layer CNN	**44.71%**	**37.78%**
1-Layer LSTM	**54.02%**	**38.46%**
3-Layer LSTM	**54.85%**	**37.09%**

Table 2: Model Performance

As described in the Implementation section of our project, we have implemented four distinct models, each showcasing varied performance levels. Our analysis has led us to select the 1-layer CNN as our final model for prediction, primarily due to its highest test accuracy of 40.38%. Despite achieving the best results among the models tested, the test accuracy for all models caps at around 40%. This ceiling is influenced by multiple factors: Firstly, the limited scope of our dataset, which, while extensive, does not cover the vast vocabulary necessary to fully capture every nuanced emotional expression. This limitation is crucial as emotional recognition in text is inherently complex due to the subjective nature of emotional labeling. Our dataset includes 28 emotion labels, and the slight discrepancies between similar emotions can affect the precision of the model predictions.

Additionally, the lower performance of the multi-layer models, such as the 2-layer and 3-layer configurations, suggests that these models are overly complex for the scope of our task. The additional layers in these models may lead to a loss of important information during processing, as each successive layer can potentially abstract away critical details that are essential for accurate emotion detection.

Furthermore, the LSTM models did not perform as expected, which can be attributed to the redundancy created by the encoder transformer. The transformer enriches the word embeddings with detailed contextual information relative to all other words in the sentence. This rich contextual embedding provided upfront diminishes the LSTM's unique advantage in handling sequential and recurrent information, which traditionally aids in capturing temporal relationships within the text. Thus, the specific benefits of LSTMs are less pronounced when the initial data processing with the transformer already encapsulates extensive contextual details.

6 Conclusions

In conclusion, our analysis of various model architectures for our project indicates that CNN-based architectures are more suited to the task of emotion detection in text. Increasing the depth of the models, as seen with the 2-layer and 3-layer configurations, does not yield improvements in performance, suggesting that a simpler model structure is preferable for this specific application. Among the encoder models tested, 'albert-base-v2' demonstrated superior performance, making it the optimal choice for our transformer encoder due to its balance of accuracy and training efficiency.

Furthermore, a significant limitation affecting all models is the size of the training dataset. Our findings clearly show that the dataset, while comprehensive to a degree, is not large enough to cover the extensive range of emotional expressions and vocabulary necessary for high-level performance in emotion detection tasks. The models' ability to predict nuanced emotional tones would likely improve with an expansion of the dataset, allowing for more comprehensive training and better generalization to unseen data. Expanding the dataset would provide a more robust foundation for the models to learn from, potentially increasing accuracy and the effectiveness of the sentiment analysis performed by our model.

References

[1] Dana Alon and Jeongwoo Ko. Goemotions: A dataset for fine-grained emotion classification, October 28 2021. Google Research Blog.

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

Appendix

Emotion Class ID	Emotion
0	Admiration
1	Amusement
2	Anger
3	Annoyance
4	Approval
5	Caring
6	Confusion
7	Curiosity
8	Desire
9	Disappointment
10	Disapproval
11	Disgust
12	Embarrassment
13	Excitement
14	Fear
15	Gratitude
16	Grief
17	Joy
18	Love
19	Nervousness
20	Optimism
21	Pride
22	Realization
23	Relief
24	Remorse
25	Sadness
26	Surprise
27	Neutral

Table 3: Emotion Class ID Chart