Sentiment Analysis on Social Media: comparison between LSTM and LLMs

Sungkyun Yoo

Overview

The project investigates sentiment analysis techniques for social media posts using both traditional deep learning (LSTM) and modern transformer-based approaches. The study aims to compare and choose the best performing model to understand human behaviour and emotions through social media content analysis, with potential applications in Large Language Model training. The key findings are: Class imbalance significantly impacted model performance, PEFT with LoRA for LLMs showed promising results despite data limitations, and Computational efficiency favoured simpler LSTM approaches for smaller datasets. The insights are the trade-offs between traditional deep learning and transformer-based approaches for sentiment analysis, particularly in scenarios with limited data and class imbalance issues.

1. Introduction

A great scale of development of the new technologies made it easier for individuals to share their thoughts and emotions on the Internet. Due to this, understanding human behaviours and emotions has become a key to understanding human behaviour and society. Sentiment analysis on the user generated posts on the social media platforms became one of the promising techniques to understand psychology [1]. While drawing the emotions from the text, there needs to be a depth of analysis rather than labelling them as simple - positive, neutral, negative [5, 8]. Not only for the purpose of social studies, dataset labelled with sentiment can be used for training the Large Language Models [10]. After fine-tuning, the model can be further tuned using reinforcement learning to generate completions with specific sentiment characteristics or avoid generating hateful or negative content.

2. Data

The dataset contains text of posts by each user from the three platforms, Instagram, Facebook, and Twitter. The numbers for each platform are 258, 231, 243, respectively. This data was retrieved from 15th of May 2010 to 22nd of October 2023, and labelled with the corresponding sentiments [Figure 1]. The data was mostly collected from the year 2023 compared to other earlier years. The imbalance timestamp of data can be ignored for the objective.

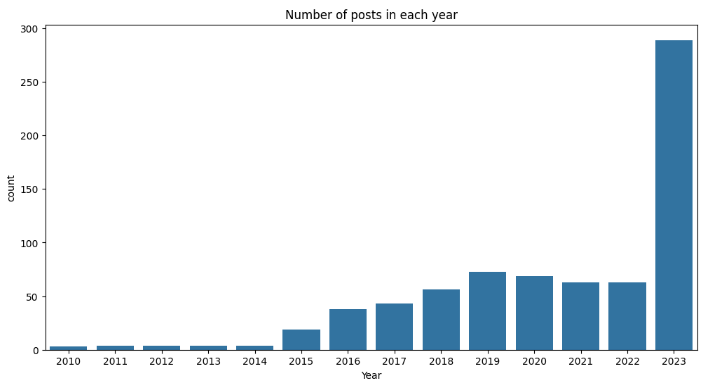

Figure 1: Number of posts on social media each year

Since there were 192 different sentiments, the data was modified with the top 10 sentiments which were the most expressed [Figure 2]. From the figure, we can assume there will be a poor performance on any NN model due to the class imbalance problem.

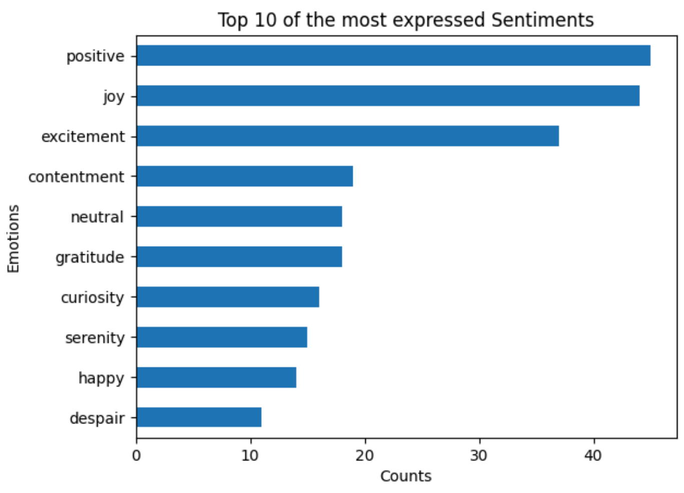

Figure 2: Number of top 10 sentiments on social media

3. Model

The report employs two distinct approaches to sentiment analysis: traditional LSTM networks and modern transformer-based models. Examples of these are two different language experts - one who learned through years of sequential reading (LSTM) and another who can look at entire texts at once (Transformer).

3.1. LSTM

To handle the natural language from the post and the sentiment, Natural Language Toolkit was used for a number of reasons. If the text contains any url, punctuation, or stopwords, it can filter out those words from the text. By using Lemmatizer, we can extract the root of the words. These can be regarded as the basic cleaning method for processing Natural Language Processes. Also, due to containing emojis in the text, converting the emoji to a text was used with the demojize method. To train and test the models, 80% of the dataframe split into the training data and the rest as the test data.

Since the data now contains 10 sentiments with the different size of each sentiment variable, for the first approach, a simple LSTM for each sentiment was applied to predict each corresponding sentiment. This simple model was integrated with an embedding layer. This integration helps convert the words into dense vectors with the corresponding embedding dimension, which can be processed in the future. The next implemented layer is LSTM, a variant of Recurrent Neural Network, which helps the model remember the information from the long sequences [12]. The next layer, Dropout, was used to prevent overfitting. The layers of activation functions were used to reduce the compute cost and avoid vanishing gradient problems, and binary classification for each sentiment with the Rectified Linear Unit, and Sigmoid, respectively. Then this model was compiled with the adam optimizer to deal with the gradients and binary cross-entropy loss function due to the sigmoid output.

The first data used for sentiment was positive with 36 samples and the whole data contains numerous non-positive data. Hence, weights on each variable were calculated, which are 0.5328 and 8.125, for non-positive and positive, respectively. After tuning the hyperparameters, the best model produced the training accuracy and test accuracy with 97.61% and 92.52%, respectively. The weighted average F1-score produced 93%. This seems to be a good result, however, from Figure 3, due to the suffer of class imbalance problem, the model cannot predict the minority class, positive, well enough.

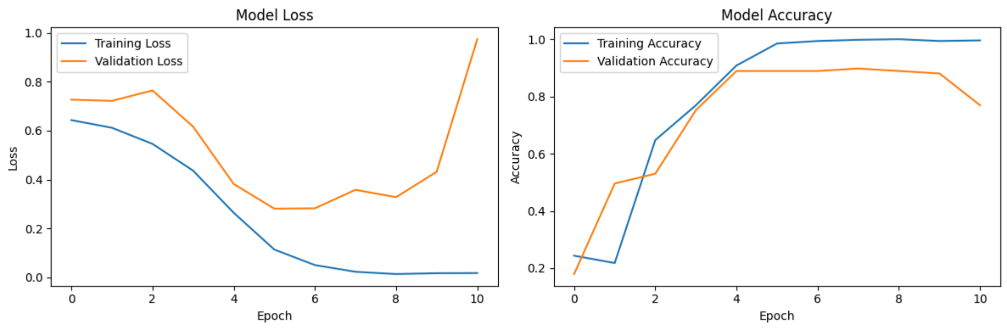

Figure 3: 'positive' Model Loss and Accuracy history

With the same model, there was one more experiment conducted with a sentiment, joy. With a similar reason as positive, class weighting method was integrated and hyperparameters for the model were tuned. The model produced 93.16% and 92.52% on the training accuracy and testing accuracy, respectively. While comparing this model with joy to positive, the weighted average f1-score was 90% and the loss of the model went through decrease in loss in general [Figure 4]. Also, the value of binary cross-entropy loss of this model went close to the acceptable range around 0.44 minimum on validation.

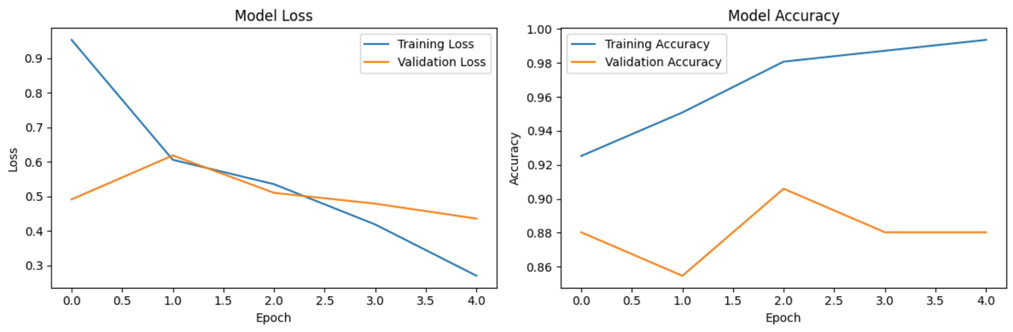

Figure 4: 'joy' Model loss and accuracy history

Though the dataset suffers a class imbalance problem, implementing a second model to predict the top 10 sentiments was conducted. This second contains an embedding layer, Bidirectional LSTM layer, two dropout layers between a Rectified Linear Unit layer, Sigmoid layer, and ends with a Softmax layer. Unlike normal LSTM, Bidirectional LSTM captures sequences in both a forward direction and a backward direction. By processing in two directions, the model can produce better performance on higher dependencies. The Softmax function calculates probability distribution for the classes which means this can be applied for multi-class classification. Since the model is going to be used for multi-class classification, it was compiled with a categorical cross-entropy loss function. The performance of the model is shown in Figure 5. The model performs well on detecting true negatives but not on true positives, which can be seen from the macro average score. Applying weights on the corresponding variables made the performance more accurate. However, as discussed above, Figure 6 shows that the loss on validation dataset was relatively high due to the class imbalance.

Sentiment	F1 score	Macro avg. score	Weighted avg. score
positive	0.96	0.49	0.94
joy	0.94	0.48	0.91
excitement	0.95	0.49	0.93

contentment	0.97	0.49	0.96
gratitude	0.97	0.49	0.96
neutral	0.97	0.49	0.96
curiosity	0.98	0.49	0.97
serenity	0.98	0.49	0.97
happy	0.98	0.49	0.97
despair	0.99	0.50	0.98

Figure 5: A table for metrics on top 10 sentiments

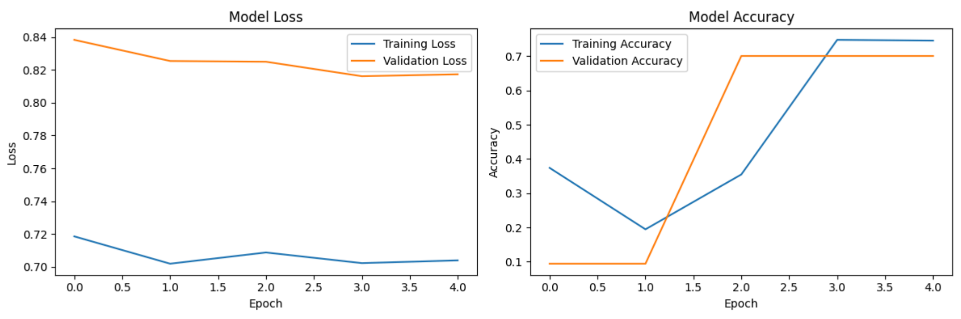

Figure 6: Bidirectional LSTM model loss and accuracy history

3.2. Transformers

For a pre-trained base model using transformer architecture, Google's FLAN-T5 was used. FLAN stands for Fine-Tuned Language Net, which means a set of instructions to fine-tune different models [13]. This model is a variant of the Text-to-Text Transfer Transformer model. In other words, This model was expanded and built upon the T5 architecture. Its base model T5-base has twice the size of the BERT, 220 million parameters compared to 110 million parameters [11]. FLAN-T5-base, learned 250 million parameters, was fine-tuned with 473 datasets across 146 task categories: T0-SF, Muffin, CoT(reasoning), and Natural Instructions [14]. Since this dataset suffers from a class imbalance problem, the model was fine-tuned, if the model makes a great performance, it would be valuable to compare with BERT, which is an encoder-only model. However, the result did not give any further insight to compare other Large Language Models.

With 90:10 split for a dataset into train and test datasets, the model was tested without any prompt engineering. As the model was not trained for a sentiment analysing task, the model could not produce any sentiment. There were 3 different training methods integrated, In-context learning, Full-Fine tuning, and LoRA with PEFT.

In-context learning is giving the *n* number of examples in the prompt to improve the model to generate more accurate completion. With one-shot inference, the model started to produce a sentiment, however, despite four-shot inference, the model could not produce an accurate sentiment. This may be caused by the class imbalance problem or the size of the data was not large enough.

Before integrating Instruction Full-Fine tuning, the dataset was pre-processed with adding instruction prompts, tokenizing, and padding. The importance of padding is that the length of sequences in each text are different, so that the Neural Network model can be trained with a consistent dataset shape. In order to calculate the performance for a model ROUGE metric was used. This metric can be interpreted like the equations below [6].

$$\text{ROUGE-1 Recall} = \frac{\text{unigram matches}}{\text{unigrams in reference}} \qquad \text{ROUGE-L Recall} = \frac{\text{LCS(output, reference)}}{\text{unigrams in reference}}$$

$$\text{ROUGE-1 Precision} = \frac{\text{unigram matches}}{\text{unigrams in output}} \qquad \text{ROUGE-L Precision} = \frac{\text{LCS(output, reference)}}{\text{unigrams in output}}$$

$$\text{ROUGE-1 F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \qquad \text{ROUGE-L F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The range of the metric is from 0 to 1, where 0 represents the lowest accuracy and 1 represents the maximum accuracy. Full-Fine tuning on the dataset took 3:54:06 of runtime, however, the ROUGE-1 score for unigram did not improve. This score for the original model was 0.23 which is higher than the ROUGE-1 score Instruction model produced, 0.2. The reasons why the accuracy of the fully fine-tuned model was lower than that of the original model are the class imbalance and relatively small size of data corresponding to each sentiment as mentioned above. If the dataset met the requirements for Large Language Models, the full-fine tuning method is expected to generate the maximum accuracy in theory. However, the computational cost would be the expected problem for this method.

One of the most prominent methods is Parameter-Efficient Fine Tuning (PEFT). The key difference of this method than the traditional fine-tuning method is updating the small subset of the parameters and freezing the other LLM weights which then be memory efficient and giving the ability to deploy multiple tasks [15]. For the reparameterization method for PEFT, Low-Rank Adaptation was used. For efficiency, the LoRA method decomposes weights into two smaller rank matrices and trains those instead of the full model weights [8]. Applying the PEFT configuration, the model training runtime was 2:11:01 which was clearly computationally less expensive. The ROUGE-1 score for the PEFT model improved from 0.23 to 0.25, as the benefit of fine tuning the model. The consequence for the fine-tuning method was expected as it represents from Figure 7. However, the dataset was not large enough to perform well on LLM, as the result shows.

Fine-tuning type	ROUGE-1 score	ROUGE-L score
Original	0.23	0.2
Full-fine tune	0.2	0.2
PEFT	0.25	0.25

Figure 7: A table for ROUGE-1 (uni-gram) scores for each fine-tuning method

4. Discussion

The dataset suffers from the class imbalance problem. Due to the relatively significant different sizes of each label, any of the models performed well on detecting true negatives, such as non-positive labels, but not reliable performance on positive labels. When it comes to the labelling technique, since there were 192 different sentiments, NRC-VAD Lexicon [16] or multi-labelling could be used to avoid mislabelling. Not only with the labelling technique, though the size of the whole dataset was sufficient to train LSTM models, but this dataset can improve the size of each sentiment for training LLM.

If the size of the dataset was large enough, applying sampling methods can be considered to avoid the class imbalance problem. SMOTE, random sampling, and ADASYN methods can be considered. SMOTE creates new synthetic examples for the minority class, uses k-Nearest Neighbors method, and interpolates the examples to the original dataset [17]. ADASYN is an adaptive method of SMOTE, using density distribution to determine more generated synthetic examples [18].

5. Conclusion

Identifying the class imbalance of the dataset should be the first procedure while considering the dataset. If the dataset struggles with the issue, sampling methods can be considered to rectify or minimise any prominent poor performance. Since the size of the dataset was not large enough, the performance of LLMs did not produce any acceptable accuracy. However, applying PEFT made a noticeable improvement on accuracy. Applying PEFT with LoRA reparameterization can make a more positive outcome when all the requirements for the dataset are met. When the dataset is not large enough, rather than applying hidden layers on LSTM or integrating LLMs, a model with a simple LSTM layer generates a more relatively accurate outcome. Since the training on LLMs is computationally expensive, applying simple LSTM to train a model generates high accuracy on sentiment analysis with being more computationally efficient, when the dataset is not large enough.

6. References

[1] Rohitash Chandra, Aswin Krishna (2021). COVID-19 sentiment analysis via deep learning during the rise of novel cases. Retrieved from: https://arxiv.org/pdf/2104.10662
[2] Ashish Vaswani et al. (2017). Attention is all you need. Retrieved from: https://arxiv.org/pdf/1706.03762
[3] Jimmy Lei Ba et al. (2016). Layer Normalization. Retrieved from: https://arxiv.org/pdf/1607.06450
[4] Kaiming He et al. (2016). Deep Residual Learning for Image Recognition. Retrieved from: https://arxiv.org/pdf/1512.03385
[5] Amit Goldenberg et al. (2020) Collective Emotions. Retrieved from: https://www.hbs.edu/ris/Publication Files/The Psychology of Collective Emotions_6308f014-3486-457c-88b6-b38db3340fb4.pdf
[6] Chin-Yew Lin, Eduard Hovy. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. Retrieved from: https://aclanthology.org/W04-1013.pdf
[7] Tom B. Brown et al. (2020). Language Models are Few-Shot Learners. Retrieved from: https://arxiv.org/pdf/2005.14165
[8] Edward Hu et al. (2021). LoRA: Low-Rank Adaptation of Large Language Models. Retrieved from: https://arxiv.org/pdf/2106.09685
[9] David Garcia et al (2021). Social media emotion microscopes reflect emotional experiences in society at large. Retrieved from: https://arxiv.org/pdf/2107.13236
[10] Tony Diana, (2022). Using sentiment analysis to reinforcement learning: The case of airport community engagement. Retrieved from: https://www.sciencedirect.com/science/article/pii/S0969699722000497
[11] Colin Raffel (2022). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Retrieved from: https://arxiv.org/pdf/1910.10683
[12] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," in IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 157-166
[13] Jason Wei et al. (2021). Finetuned Language Models Are Zero-Shot Learners. Retrieved from: https://arxiv.org/pdf/2109.01652
[14] HyungWon Chung et al. (2022). Scaling Instruction - Finetuned Language Models. Retrieved from: https://arxiv.org/pdf/2210.11416
[15] Lingling Xu et al. (2023). Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment. Retrieved from: https://arxiv.org/pdf/2312.12148
[16] Saif Mohammed. (2018). Word Affect Intensities. Retrieved from: https://nrc-publications.canada.ca/eng/view/ft/?id=baca6153-1b13-4e0b-9a1c-280150938712
[17] Nitesh Chawla et al. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Retrieved from: https://arxiv.org/pdf/1106.1813
[18] Haibo He, Yang Bai, E. A. Garcia and Shutao Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, 2008, pp. 1322-1328

7. Appendix

Github link for this report: https://github.com/JulianSKYOO/transformer
Dataset retrieved from:
https://www.kaggle.com/datasets/kashishparmar02/social-media-sentiments-analysis-dataset/data