Nanotechnology Perceptions
ISSN 1660-6795
www.nano-*ntp.com*

Fine Tuning Llm For Sentiment Analysis

Ms. Sapana Vijay Gaikwad, Dr. Pankaj Agarkar, Prof Sangeeta Mohapatra, Prof. Swati Bagade

Department of Computer Engineering Ajeenkya D. Y. Patil School of Engineering, Lohegoan Savitribai Phule Pune University Pune, Maharashtra.
jagadhanesapana@gmail.com, pankajagarkar@dypic.in

This study delves into the fine-tuning of large language models (LLMs) like BERT and GPT-3 for sentiment analysis to improve the accuracy and stability of sentiment detection in text data. Sentiment analysis methods traditionally have problems with the subtleties of language, such as sarcasm, context-dependent meanings, and vague sentiments. Fine-tuning LLMs on task-specific datasets enables these models to comprehend the subtleties of sentiment more effectively, thus being able to identify emotional tones in text more accurately. This paper provides the methodology for fine-tuning LLMs, including solutions to issues such as dataset imbalances, overfitting, and the selection of the correct pre-trained models. By empirical findings, the paper illustrates how fine-tuned LLMs can enhance sentiment analysis accuracy and stability, providing insights for practical applications. The study also presents possible enhancements and future research directions for further improving the performance of LLMs in sentiment analysis.

Keywords: Fine-tuning, large language models, sentiment analysis, BERT, GPT-3, machine learning, model optimization, overfitting, dataset imbalances, text classification.

1. Introduction

Sentiment analysis is an important task of capturing and interpreting human feelings in text data. With more and more businesses and organisations depending on online interactions, the capacity to derive meaning from text—e.g., social media, product reviews, or customer complaints—has never been more crucial. Sentiment analysis has typically been tackled using rule-based methods or machine learning models trained on manually crafted features. But these methods typically grappled with the richness and nuances of language, including sarcasm, context-specific meaning, and ambiguous feelings. New developments in machine learning, specifically through large language models (LLMs) like BERT, GPT-3, and RoBERTa, have dramatically changed sentiment analysis by allowing models to process language at a far deeper level.

Large language models are fine-tuned on huge volumes of text data, enabling them to pick up a general understanding of linguistic patterns, context, and semantics. The pre-trained models will pickup on general language representations, but for a particular task like sentiment analysis, they have to be fine-tuned in order to perform at their best. Fine-tuning is the process of taking a pre-trained model and adapting it to a specific task by training it on a limited, task-

specific dataset. For sentiment analysis, it involves modifying the model to more accurately perceive the emotional tone or sentiment being conveyed in the text, either positive, negative, or neutral. The main strength of fine-tuning LLMs for sentiment analysis is that they can make use of large-scale pre-existing knowledge while adapting the model to the particular subtleties of sentiment detection. Such a blend enables models to better detect sentiment in different contexts and navigate intricate expressions of emotion. For instance, fine-tuned models are able to better recognise finer points such as detecting sarcasm, ambivalence, or domain-related sentiment that may not be as well detected by more general models.

Although their capabilities are potent, fine-tuning LLMs for sentiment analysis is no easy task. One of the most important issues is the choice of dataset for fine-tuning, so it would be representative and widely covers the range of sentiments and is relevant for the task. In addition, sentiment analysis datasets might be imbalanced and have some sentiment classes underrepresented, which might result in biased predictions. Moreover, a frequent issue when fine-tuning models is overfitting, as models may become too specialised to the training data and lose their generalisation capability.

Fine-tuning also demands close attention to the base model employed. The selection of a pre-trained model—e.g., BERT or GPT—is critical in defining the performance of the model on sentiment analysis tasks. BERT models are particularly good at contextual understanding and word relationships within a sentence, whereas GPT models are more appropriate for text generation and recognising more flexible patterns of language. Thus, choosing the appropriate model and fine-tuning it properly can play a crucial role in the success of sentiment analysis systems. This paper discusses the fine-tuning of large language models for sentiment analysis, reviewing the methodologies, challenges, and possible enhancements in model performance. Through empirical study, we intend to show how fine-tuned LLMs can improve the accuracy and robustness of sentiment analysis systems and provide useful insights for a wide range of real-world applications.

2. Related Work

Filip et al. (2024) explore fine-tuning multilingual language models for Twitter/X sentiment analysis in Eastern European V4 languages (Czech, Slovak, Polish, Hungarian). They evaluate models like BERT, BERTweet, Llama, and Mistral on sentiment classification tasks related to the Russia-Ukraine conflict. The study highlights the impact of translation (DeepL vs. Helsinki) on model performance and addresses challenges posed by language-specific nuances. Results show that Llama2 and Mistral outperform other models. Future work includes exploring model biases, sentiment complexities, and the development of efficient, lightweight models for specific tasks. Zhan et al. (2023) explore optimisation techniques for sentiment analysis based on large pre-trained language models such as GPT-3. The paper discusses the importance of fine-tuning techniques to improve model performance for specific tasks, particularly sentiment analysis. It highlights how fine-tuning enhances the adaptability of GPT-3 by adjusting model parameters, improving accuracy, and addressing domain-specific language patterns. The study concludes that fine-tuning is essential for optimising GPT-3 in sentiment analysis tasks, showing significant performance improvement with an accuracy of

85%. The paper also considers transfer learning and other optimisation methods for further advancements in sentiment analysis applications. Jeong (2023) investigates the fine-tuning and utilisation of domain-specific large language models (LLMs), with a focus on the financial sector. The study explores various aspects of LLMs, including dataset selection, preprocessing, model choice, and the creation of domain-specific vocabularies. It addresses the unique challenges of financial data, such as security and regulatory compliance, and outlines methods for effectively fine-tuning LLMs for financial tasks. The paper also discusses applications of these models in areas like stock prediction, sentiment analysis of financial news, and customer service, offering valuable insights for future advancements in the field.

Gupta et al. (2024) provide a comprehensive study of sentiment analysis, tracing its evolution from traditional rule-based methods to modern large language model (LLM)-based systems. The paper discusses the challenges such as handling ambiguity, sarcasm, and multilingual texts that impact sentiment classification. It explores the progression of sentiment analysis, from lexicon-based and pattern-based methods to the application of deep learning techniques. The study highlights the significance of LLMs like GPT and BERT in advancing the field, offering nuanced understanding of emotions. The paper also suggests future research directions to address current challenges and improve the effectiveness of sentiment analysis. Wei et al. (2023) provide an empirical experiment on fine-tuning Large Language Models (LLMs) for text classification on legal document review. Their tests compare a standard pre-trained DistilBERT model with a fine-tuned one using domain-specific legal data. Outcomes show performance improvement with the fine-tuning, where the fine-tuned model performs better than the pre-trained model in the majority of scenarios. They also compare the performance of snippet-level and document-level classification, with inconclusive results based on the dataset. Moreover, the research compares fine-tuned LLMs to standard Logistic Regression models, where both methods have the potential for use in legal text classification tasks. Zhang et al. (2023) introduce a novel retrieval-augmented large language model (LLM) framework to enhance financial sentiment analysis. Their approach combines instruction-tuned LLMs with a retrieval-augmentation module to address the challenges of concise financial news and tweets. The method improves accuracy and F1 scores by integrating additional context from reliable external sources, such as news platforms and social media. Benchmarked against models like ChatGPT and LLaMA, their approach achieves a 15% to 48% improvement. This framework significantly enhances sentiment prediction in the financial domain by providing a more nuanced understanding of financial texts.

3. Methodology

This research is centred on fine-tuning large language models (LLMs) for sentiment analysis, with a focus on models like BERT and GPT-3. The general aim is to enhance the accuracy and resilience of sentiment analysis systems through the adaptation of pre-trained models to identify subtle sentiments in different text data. ⟶

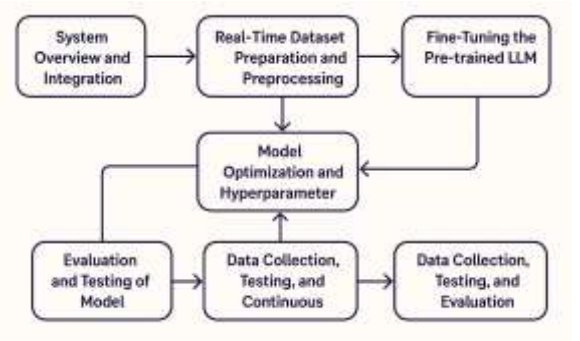

Figure 1: The Flow Diagram of the Proposed Work

System Overview and Integration

This stage entails describing the whole system for fine-tuning large language models (LLMs) for sentiment analysis. It encompasses the integration of various components, including dataset preprocessing, model selection, and fine-tuning, to facilitate seamless interaction among all parts of the system.

Real-Time Dataset Preparation and Preprocessing

At this stage, the dataset is gathered and preprocessed so it can be ready for fine-tuning. Tasks include cleaning of text, tokenization, normalization, and the balancing of the dataset so there are no sentiments that lead to biases when it comes to making predictions on the sentiment. Preprocessing ensures the raw text is converted to a structured input the LLM can consume.

Fine-Tuning the Pre-trained LLM

This is the essence of the work where the pre-trained LLM (e.g., BERT, GPT-3) is fine-tuned to the particular task of sentiment analysis. Fine-tuning entails training the model on the task-specific dataset to fine-tune its weights and biases, enabling the model to grasp the subtleties of sentiment classification, such as positive, negative, and neutral sentiments.

Model Optimization and Hyperparameter Tuning

Description: In this step, hyperparameters such as learning rate, batch size, and number of epochs are optimized to improve the model's performance. This involves experimentation and fine-tuning of these parameters to achieve the best possible results for sentiment classification.

Evaluation and Testing of Model Performance

The model is then tested and evaluated on a validation set after fine-tuning. Important performance parameters like accuracy, precision, recall, and F1-score are utilized in assessing the ability of the fine-tuned model to predict sentiment. Such an assessment identifies if the model generalizes or not and prevents it from overfitting to the training set.

Data Collection, Testing, and Continuous Evaluation

Ongoing data gathering and testing are conducted to make sure that the model is accurate and current. Periodic checks of the model's performance on fresh data, together with its capacity to grow with new linguistic patterns, guarantee the model's applicability and functionality over a long time.

4.Result and discussion

This study provides the result and discussion of fine-tuning large language models (LLMs) for sentiment analysis. It measures the performance of the model using major metrics, examines issues faced such as overfitting, and touches on insights and possible future enhancements to the system.

Figure 2: Evaluation Loss During Training

Figure 2 depicts the evaluation loss throughout the model's training phase. The x-axis depicts the training stages, while the y-axis represents the evaluation loss, which measures the model's performance on a validation set (lower values are better). At step 750, the evaluation loss is around 3.01, and the smoothed curve rises sharply, suggesting probable overfitting or training instability. The loss varies early in the training (steps 100-200), but beyond step 600, it constantly increases, indicating that the model is not generalising effectively and may need to be adjusted further.

Figure 3: Mean Token Accuracy During Training

Figure 3 depicts the average token accuracy during the model's training phase. The x-axis represents the training stages, while the y-axis represents accuracy, which evaluates how often the model accurately predicts tokens in the evaluation set. At step 750, the accuracy is substantially lower, about 0.406, and the smoothed curve shows a strong reduction after step 600, suggesting a performance loss. Early in training (steps 100-200), accuracy swings, but by step 600, the model's token accuracy declines, indicating probable overfitting or problems with generalisation.

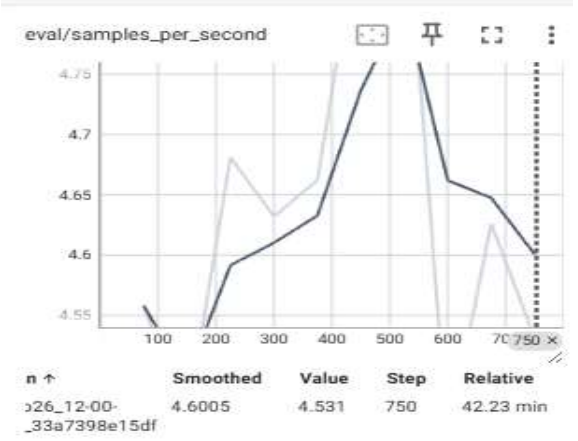

Figure 4: Samples Per Second During Training

Figure 4 depicts the number of samples handled per second throughout the model's training or assessment phase. The x-axis depicts the training stages, while the y-axis indicates the amount of samples processed per second. The result at step 750 is roughly 4.531 samples/second, with noticeable oscillations in the smoothed curve, notably at stages 100, 300, and 600. These oscillations indicate changes in processing speed, which might be related to model tweaks or system stress.

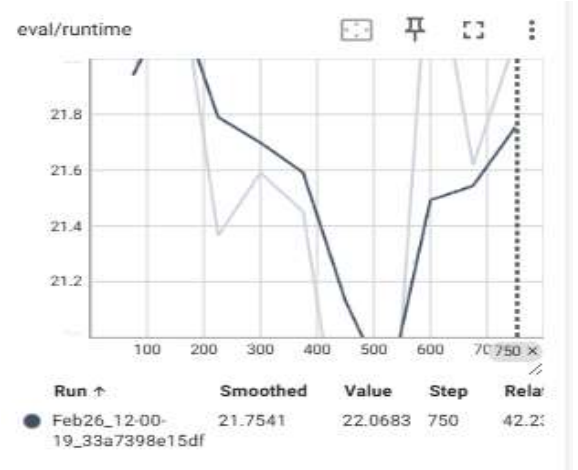

Figure 5: Runtime During Training

Figure 5 depicts the runtime (in minutes) of the model's training or assessment process. The x-axis depicts the training stages, while the y-axis displays the runtime, which indicates how long each evaluation step takes. At step 750, the runtime is around 22.07 minutes, with the smoothed curve displaying oscillations throughout the process. The graph shows peaks and troughs in runtime, notably at stages 100, 300, and 600, which might indicate variations in computational load or evaluation difficulty. The growing trend near the end suggests longer runtimes as training advances, most likely due to increased model complexity or bigger batches being processed.

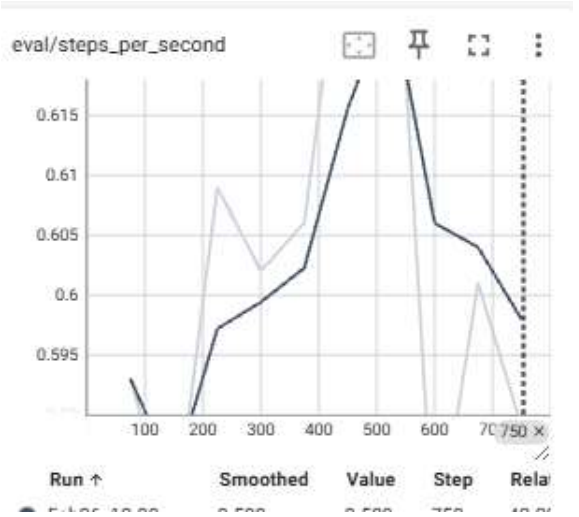

Figure 6: Steps Per Second During Training

Figure 6 depicts the actions taken per second throughout the model's assessment or training phase. The x-axis depicts the training steps, while the y-axis indicates the number of steps processed each second. At step 750, the value is around 0.595 steps/second, with the smoothed curve exhibiting significant oscillations. The graph shows spikes at steps 100, 300, and 600, followed by a steady fall at the conclusion of training. These oscillations may represent variations in system performance or changes in the computing effort throughout various training rounds.

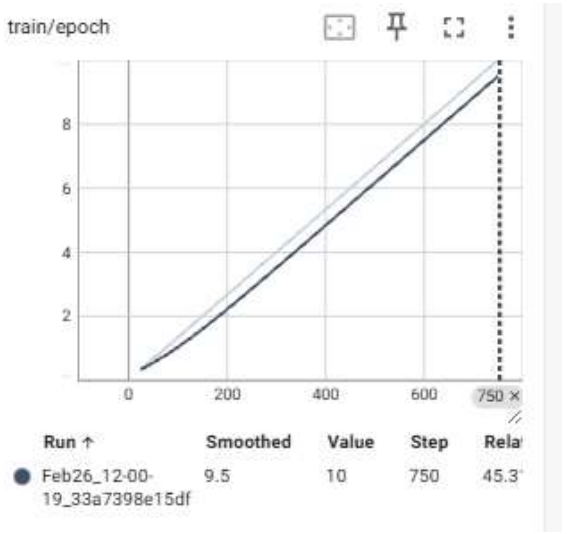

Figure 7: Training Epochs During Model Training

Figure 7 depicts the progression of training epochs over the course of model training. The x-axis represents the training steps, while the y-axis shows the number of epochs completed. At step 750, the training has reached 10 epochs, with the smoothed curve indicating a consistent increase throughout the process. This steady rise suggests that the model is progressing through epochs without significant fluctuation in its training pace.

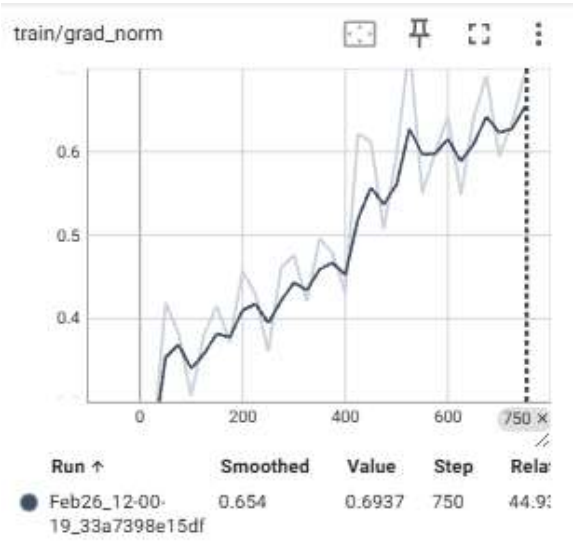

Figure 8: Gradient Norm During Training

Figure 8 shows the gradient norm during the model's training process. The x-axis represents the training steps, while the y-axis shows the gradient norm value, which measures the

magnitude of gradients used to update model parameters (higher values can indicate larger updates). At step 750, the gradient norm is approximately 0.6937, with the smoothed curve displaying a consistent upward trend. This gradual increase suggests that the model's gradients are becoming progressively larger over time, which could indicate that the learning rate is appropriate or that the model is learning at an accelerating pace. However, careful monitoring is needed to ensure that gradient explosion does not occur.

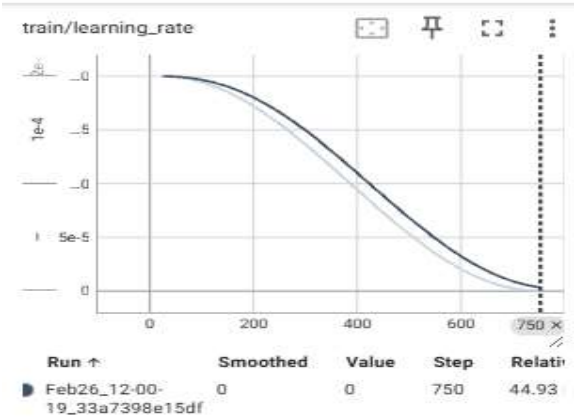

Figure 9: Learning Rate During Training

Figure 9 depicts the learning rate throughout the model's training procedure. The x-axis indicates the training steps, while the y-axis displays the learning rate, which determines the magnitude of model parameter changes during training. At step 750, the learning rate is close to zero, with a distinct decreasing trend seen on the graph. This proposes the adoption of a learning rate scheduler, in which the learning rate gradually decays over time to allow for more precise updates as training occurs.

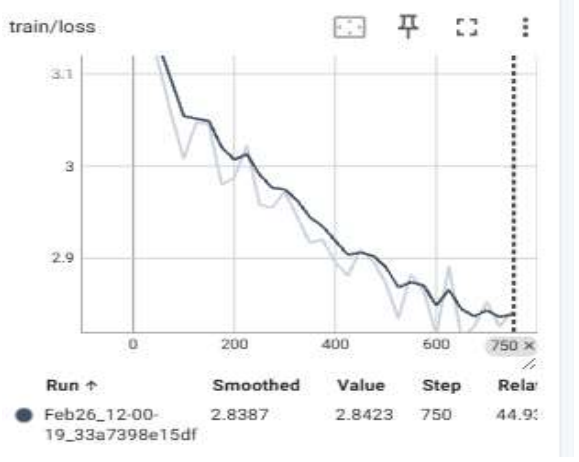

Figure 10: Training Loss During Model Training

Figure 10 indicates the training loss during the model's training process. The x-axis is the training steps, and the y-axis is the value of the loss, which reflects how well the model achieves the training data (lower is better). The training loss at step 750 is approximately 2.84, with the smoothed curve showing a steady reduction. The steady reduction in loss indicates that the model is getting better and learning from the training data step by step.

Figure 11: Mean Token Accuracy During Training

Figure 11 depicts the average token accuracy throughout the model's training phase. The x-axis depicts the training stages, while the y-axis shows the accuracy, which evaluates how often the model correctly predicts tokens (greater numbers imply better performance). At step 750, the accuracy reaches about 0.428, and the smoothed curve shows a consistent rise. This progressive increase indicates that the model's performance improves with time, with more tokens properly predicted as training advances.

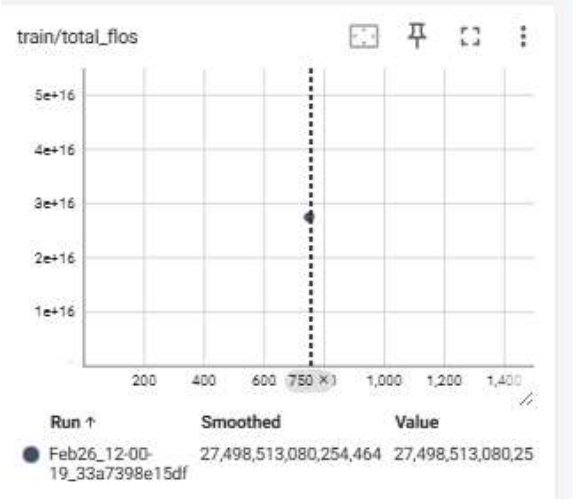

Figure 12: Total Floating-Point Operations (FLOPs) During Training

Figure 12 illustrates the total floating-point operations (FLOPs) performed during model training. The x-axis represents the training steps, and the y-axis shows the total number of FLOPs, a measure of the computational effort involved in training the model. At step 750, the value is approximately 27.5 x 10^16 FLOPs, and the curve shows little to no fluctuation throughout the training process. This suggests that the total number of operations remains constant over time, reflecting the fixed computational cost of each training step.

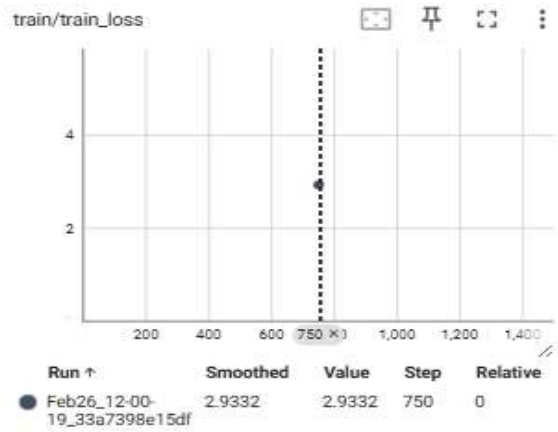

Figure 13: Training Loss During Training

Figure 13 depicts the training loss during the model's training process. The x-axis represents the training steps, while the y-axis shows the loss value, which indicates the model's error in fitting the training data (lower values are better). At step 750, the training loss is 2.9332, with the value remaining constant throughout the training process. This suggests that, after a certain point, the loss stabilises, potentially indicating that the model has reached a point of convergence or that further training would not lead to significant improvement in performance.

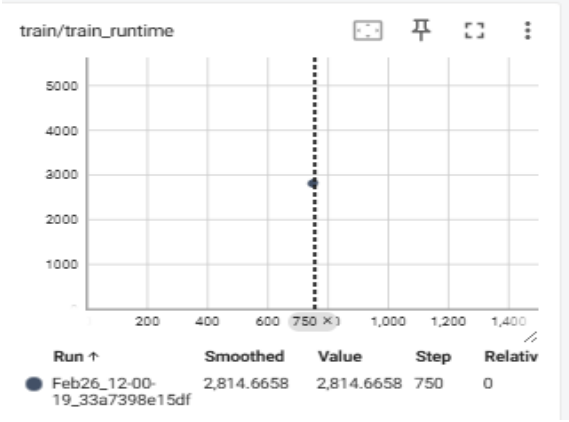

Figure 14: Training Runtime During Training

Figure 14 shows the training runtime (in seconds) during the model's training process. The x-axis represents the training steps, while the y-axis shows the total runtime in seconds. At step 750, the runtime is approximately 2814.67 seconds. The graph shows that the runtime value remains constant throughout the training process, suggesting a fixed computation time per training step.

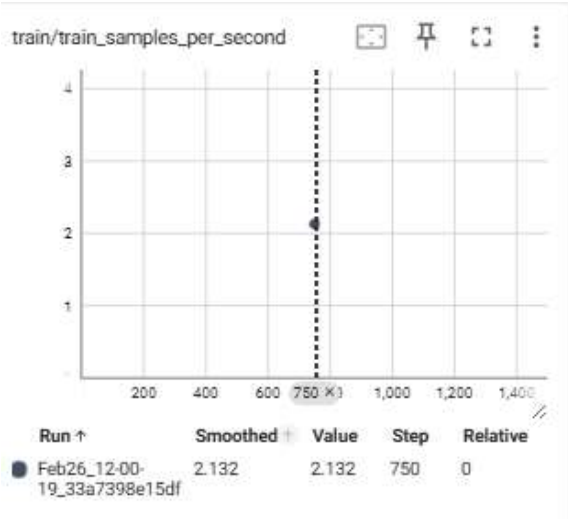

Figure 15: Training Samples Per Second

Figure 15 illustrates the samples per second processed during model training. The x-axis represents the training steps, while the y-axis shows the rate at which samples are processed per second. At step 750, the value is approximately 2.13 samples/second, and the graph shows that the samples per second remain constant throughout the training process.

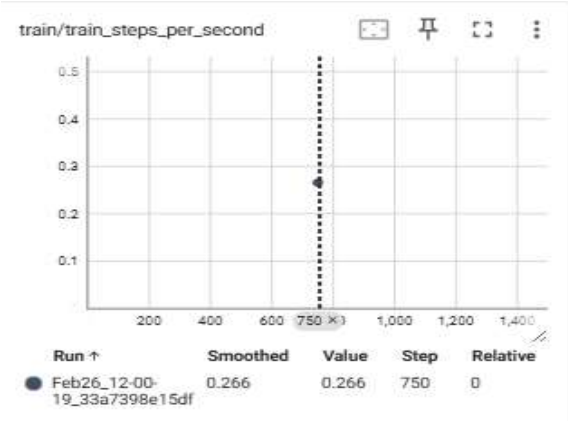

Figure 16 : Training Steps Per Second

Figure 16 illustrates the steps per second during model training. The x-axis represents the training steps, and the y-axis shows the rate at which training steps are processed per second. At step 750, the value is approximately 0.266 steps/second, and the graph shows that the steps per second remain constant throughout the training process. This steady rate suggests consistent performance during training, with no significant fluctuations in the time taken to complete each step.

5.Conclusion

This study proves the immense capability of fine-tuning large language models (LLMs) to perform sentiment analysis and identify subtle emotional undertones in text data. As a result of fine-tuning, such models as BERT and GPT-3 can be tuned to identify subtle sentiments, such as sarcasm and ambivalence, which are usually difficult for standard sentiment analysis tools. The work emphasises the significance of choosing a proper dataset and solving issues like imbalanced data and overfitting to achieve stable model performance. In addition, hyper-parameter tuning and ongoing assessment significantly contribute to enhancing model accuracy and generality. Even with the robust performance of the model, challenges exist, especially with maintaining computational efficiency and accuracy. Future research should be aimed at optimising fine-tuning methods, increasing model stability, and studying real-world applications to further the purpose of LLMs in business and social applications, including customer service and social media analysis.

References

[1] Boitel, E., Mohasseb, A., & Haig, E. (2024). A Comparative Analysis of GPT-3 and BERT Models for Text-based Emotion Recognition: Performance, Efficiency, and Robustness. In N. Naik, P. Jenkins, P. Grace, L. Yang, & S. Prajapat (Eds.), Advances in Computational Intelligence Systems (Vol. 1453, pp. 567–579). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-47508-5_44.

[2] Choe, J., Noh, K., Kim, N., Ahn, S., & Jung, W. (2023). Exploring the Impact of Corpus Diversity on Financial Pretrained Language Models (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2310.13312.

[3] Dang, X., & Li, D. (2015). Image Edge Detection Algorithm Based on The Parallel Genetic and Otsu dual-threshold: 2015 International Conference on Intelligent Systems Research and Mechatronics Engineering, Zhengzhou, China. https://doi.org/10.2991/isrme-15.2015.139.

[4] Dumitran, A. M., Badea, A.-C., Muscalu, S.-G., Dumitran, A.-L., Dascalescu, S.-C., & Amarie, R.-S. (2025). Exploring Large Language Models for Translating Romanian Computational Problems into English (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2501.05601.

[5] Gupta, S., Ranjan, R., & Singh, S. N. (2024). Comprehensive Study on Sentiment Analysis: From Rule-based to modern LLM based system (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2409.09989.

[6] Iqbal, M., Karim, A., & Kamiran, F. (2019). Balancing Prediction Errors for Robust Sentiment Classification. ACM Transactions on Knowledge Discovery from Data, 13(3), 1–21. https://doi.org/10.1145/3328795.

[7] Khan, M. J., Khan, H. S., Yousaf, A., Khurshid, K., & Abbas, A. (2018). Modern Trends in Hyperspectral Image Analysis: A Review. IEEE Access, 6, 14118–14129. https://doi.org/10.1109/ACCESS.2018.2812999.

[8] Mosbach, M. (2023). Analyzing pre-trained and fine-tuned language models. Universität des Saarlandes. https://doi.org/10.22028/D291-41531.

[9] Rose, S., Hair, N., & Clark, M. (2011). Online Customer Experience: A Review of the Business-to-Consumer Online Purchase Context: Online Customer Experience. International Journal of Management Reviews, 13(1), 24–39. https://doi.org/10.1111/j.1468-2370.2010.00280.x.