Sentiment Analysis - Fine-tuning Pre-Trained Transformer Models

(Hugging The Hugging-Face)

Doron Shapira, Ben Mali, Gal Zivoni

Abstract

In this work we will present the outcome of fine-tuning two different pretrained models for completing a sentiment analysis task. Moreover, we will go over the common and uncommon methods used to improve the model's metrics and the adjustments made to our specific task - analysis of IMDB reviews. Finally, we will present a comparison between two methods used to reduce the size of the pretrained model and discuss the recommended steps one can take to possibly obtain better results in a future study.

1. Introduction

Social media has become an integral part of human living in recent days [1]. People want to share every event happening in their life on social media. Nowadays, social media is used for showcasing one's pride or esteem by posting photos, text, video clips, etc. The text plays a vital aspect in information shared, where users share their opinions on trending topics, politics, movie reviews, etc.

Movie reviews are short texts that generally express an opinion about movies that were watched by the author of the review. These reviews play a vital role in the success of a movie effecting its gross revenues at times. People generally search for information about the movie at sites like IMDb such as movie cast, crew, reviews, and ratings. Hence, reviews play a prominent role in a person's decision between watching a movie or not. In other words, applying sentiment analysis on movie reviews helps to extract the sentiment expressed by the reviewer in his review and to give a clear understanding of his opinion about the movie. The task mainly includes preprocessing, model creation, tokenization, classification, and analysis of results. Preprocessing mainly involves removal of stop words which are scrutinize for the task of classification, while model creation involves design and implementation of a deep learning network architecture or usage of a pre-trained model.

In this paper we will present our work on IMDB Sentiment Analysis task, including methodology, results, conclusions, and discussion.

2. Methods

2.1. Exploration

Every machine learning problem requires exploration of the raw dataset to fully understand the complexity and distribution of the data. In this work, we have decided to explore the top 10 words found both in the negative and positive reviews (the dataset), as seen in the graph below:

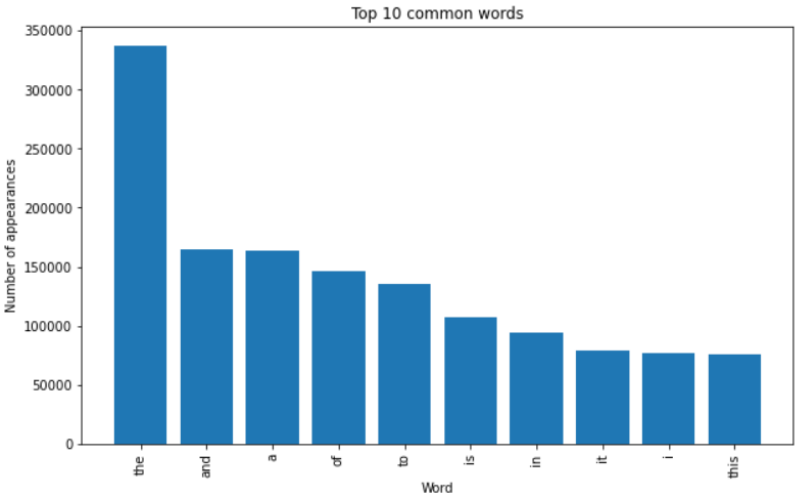

Figure 1.0 – Top 10 Common Words

We have also checked the average review length and the average rating for a review in the dataset which stood on 1263 and 5.48 accordingly (balanced set).

2.2. Preprocessing

NLP models were changed throughout the last decades. Feed Forward models were replaced by RNN's, which were later replaced by LSTMs and their enhancements, leading to LSTMs with attention heads, ending with the rise of Transformers models. Each of these models require a method to turn plain text into a vector space, most known as word embedding. The models also use a large corpus of text to learn word associations.

The primary step of training a model involves preprocessing of the data. The data on which we are focusing contains text reviews which can contain multiple combinations of letters, punctuation marks, numbers and even gibberish. Furthermore, the data is split into negative and positive reviews. In this case, it is common to remove any redundant character (i.e. punctuation marks, gibberish) and use a lower-case representation of a word, as the corpus holds only lower-cased words [2]. After removing the special characters, we have decided to further pre-process the text, using 3 different functions:

1. Minimal "N" Removal – After counting the appearances of each word (in the positive and negative reviews, separately), we have set a controllable hyperparameter – "N" – which determines the minimal number of appearances a word needs to have to not be removed from the text. Words with a number of appearances less than "N" were removed. In this work, we have set "N" to 10 appearances. Applying this function will theoretically help the model by not training on words that are uncommon and probably not contributing to the learning process.

2. Top Shared "K" Removal - After counting the appearances of each word which are found both in the positive and the negative reviews, we have set another controllable hyperparameter – "K" – which determines the maximal ranking position of the words that are needed to be removed. Words that are ranked "K" or higher (according to their shared appearances) are removed from both the negative and the positive reviews. In this work, we have set "K" to 5. Applying this function will theoretically help the model by not training on "generic" words which probably have minimal correlation with a sentiment and are more confusing than indicative (i.e., "to", "the").

3. Probabilistic Removal - After counting the appearances of each word (in the positive and negative reviews, separately), we built a probability distribution by dividing the count of appearances of a word by the sum of appearances of all the words in a category (positive and negative separately).

Then, we passed on each word in each review, generated a random number from a continuous uniform distribution (0 to 1), and removed the word if the generated number was lower than the probability value of the word. Applying this function will theoretically help the training of the model by reducing the differences in the word appearances (reducing the "duplicate" population), adding randomness to the model, and consequently reducing the bias of the model towards more common words. Nevertheless, this function is more moderate compared to the other functions, as the highest probability of a word to be removed is 0.05, and if a word is removed, it is only removed from the specific review and not from the entire dataset.

2.3 Model Selection

Finetuning pre-trained models has become more and more common in solving various NLP tasks. We have decided to finetune and compare two models – XLNet and RoBERTa - which were presented as the most accurate to the IMDB Sentiment Analysis and had the most promising results in a broad comparison between different transformer models [3]. However, XLNet and RoBERTa large model versions include 340 million parameters, forcing us to use the "base-cased" versions of these models which include 110 million parameters.

XLNet is a bidirectional transformer with permutation-based modeling which was trained on BookCorpus Wiki, Giga5 ClueWeb, and Common Crawl, while RoBERTa is a BERT model without NSP which was trained on BBcorpus Wiki CC-News Stories.

2.4 Hyperparameter Tuning

Many models suffer from underfitting during the training process, emphasizing the need for tuning hyperparameters. Moreover, Accuracy and F1 scores improvements can be achieved by smartly searching a through a large space for each hyperparameter using random search [4]. In our work, we have used an easy to set up package called "Optuna" which performs an efficient random search on a model's hyperparameters [5]. We have performed the search on 20 different runs of each model for 3 epochs (20 different random sets of hyperparameters). Below is a table describing the hyperparameters on which we have decided to run a search, and their range:

Hyperparameter	**Min Range**	**Max Range**
Learning Rate	1e-6	5e-5
Train Batch Size	16	27
Weight Decay	1e-4	1e-2
Gradient Accumulation Steps	1	6
Warmup Steps	0	500

Figure 1.1 – Hyperparameters Searched

The range used for the learning rate and weight decay was based on a similar range used in different research, considering a low learning rate for a pre-trained model as all of the weights are trained in the process. In fact, it was claimed that the best learning rates for RoBERTa are 1e-5, 2e-5 and 5-e5 [6]. We have also decided to use a tokenization with maximum length of 200 elements (with padding). However, the max length is a hyperparameter too that can be modified to theoretically obtain better result (as higher representations contain more information on the review but require more computational power).

2.5 Model Compression

With the ongoing trend of enlarging models by expanding the size of the model's network, consequently adding more parameters into use, current models are often too big to deal with [7]. They become expensive to run due to the computational power required and due to the latency they cause, and they are very difficult to fine-tune. Model compression helps us to deal with large models by applying multiple techniques such as pruning and quantization to reduce the model size and maintain the quality of the larger model. In our work, we have decided to apply weight pruning and quantization on the XLNet model and tried to compare the model's performance with the non-compressed model's performance. We have pruned 50% of each feed-forward layer's weights that had the lowest L1-norm (which theoretically should be more effective than pruning random weights). We have also applied a quantization technique (on the non-

pruned model) for the feed-forward layers which should reduce the model's size, in which the weight's are changed to integers based on a proportional rate and from a defined scale. We have set the rate to be 0.1, and the scale to start from 0. However, for both techniques we have set the number of pruned weights of the feed-forward layer and the defined scale as controllable hyperparameters, as we believe that they might require tuning for achieving better results.

2.6 Defining Metrics

In this task, metrics and their calculation should be defined properly so we can compare between models. We have used only rated reviews (from a score between 0-10) resulting in a total of 25,000 reviews for the entire dataset. Later, we have split the entire dataset into train and validation sets (75% training, 25% validation) and have shuffled between positive and negative reviews to avoid bias. The metrics we have chosen to track are the training loss, and the validation's accuracy and F1 score. However, the quality of the model will be assessed using the validation's accuracy (based on similar research).

The Infographic below depicts a summarization of the methodology that was applied on both models:

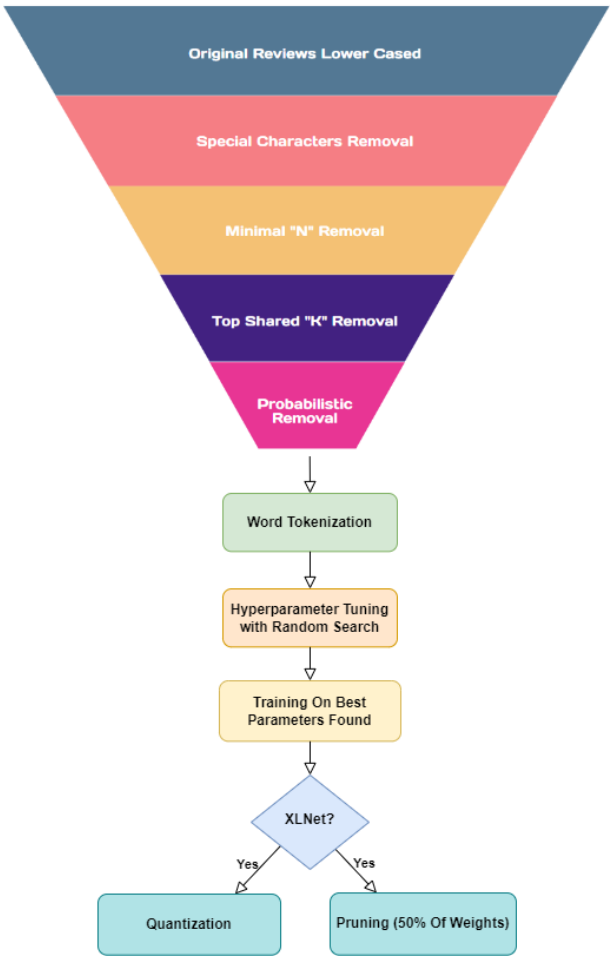

Figure 1.2 – Methodology Summarization

3. Results

3.1 Model Performance

After applying the pre-processing techniques, the average review length has been reduced to 1107 (reduced by 12%). Shortening this length allows the model to train on more samples or alternatively speed up the training process on the same number of samples, while unharming the information that can be extracted from the reviews. We can also see the top 10 words distribution has changed after removing the top 5 words in the dataset and applying the probabilistic removal, according to the next plot:

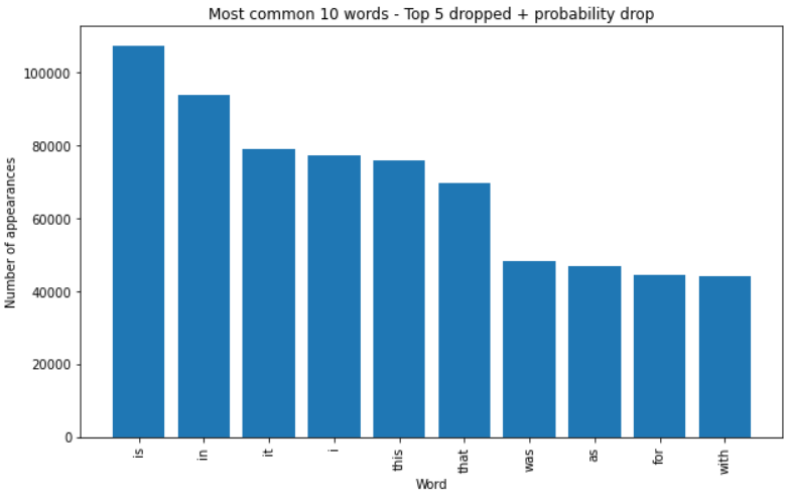

Figure 1.3 – Top 10 Words After Pre-process

Removing the words "the", "and", "a", "of" & "to" should allow our model to focus on more informative words and build correlations to the sentiment accordingly. Moreover, the probability removal has slightly affected the distribution, although not in a noticeable way. However, we do remain with less informative words such as "is" and "it", which might harm the models' chances to reach their full potential.

Later, we have performed a hyperparameter search on each model as described in the methodology section above. We have used the "wandb" library to track the relevant metrics. The following plots present the combination of hyperparameters and the derived results from using these hyperparameters, for each model:

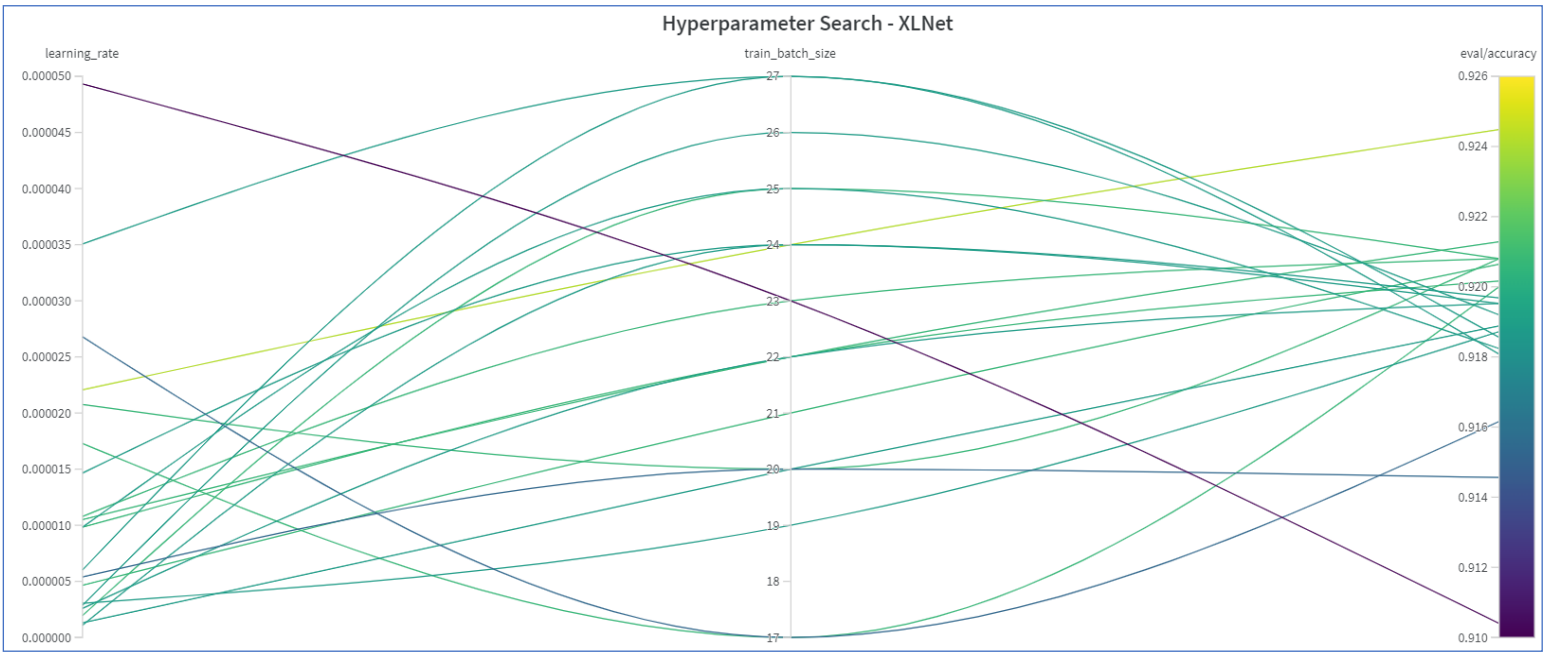

Figure 1.4 – XLNet Random Search

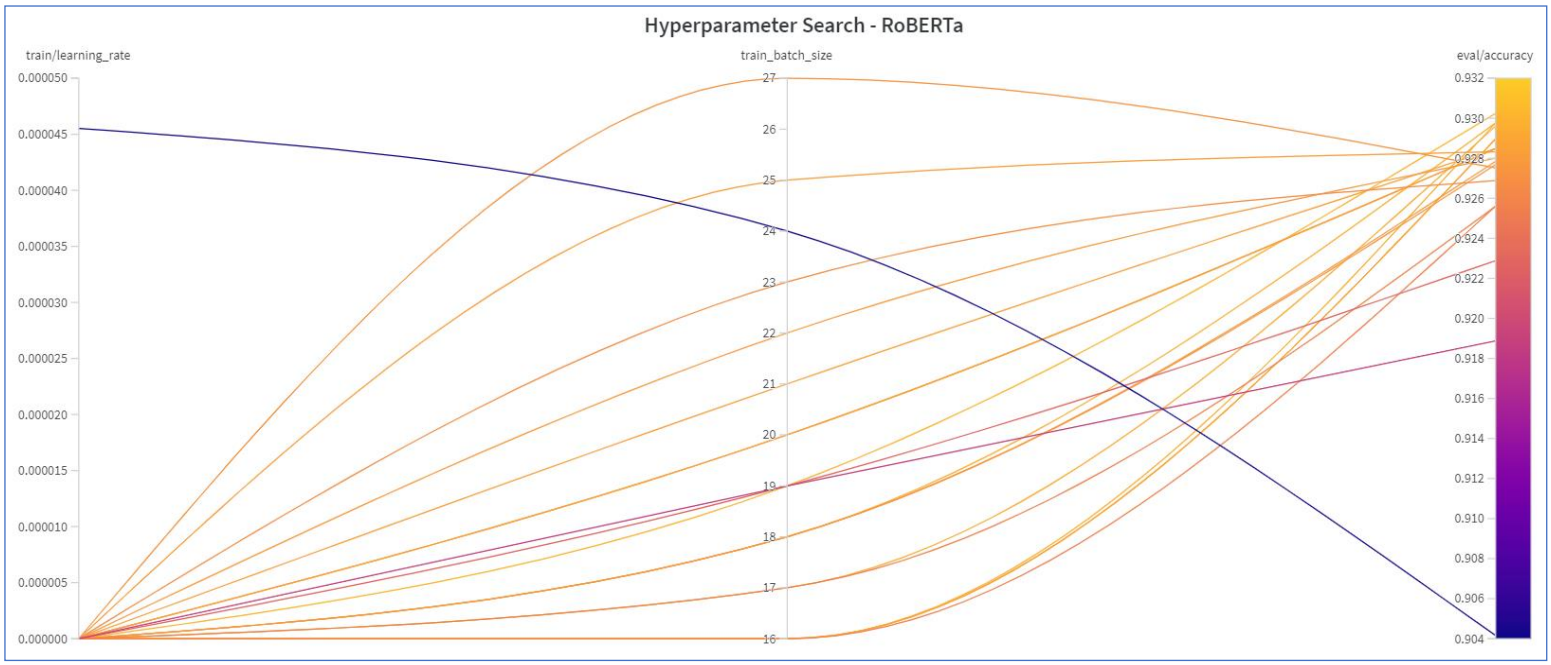

Figure 1.5 – RoBERTa Random Search

As seen in the graphs, XLNet's average accuracy over 20 different runs stands around 92%, while RoBERTa's average accuracy stand around 92.8%. However, some of the runs were overfitted (as the random search didn't allow early stopping), and the difference isn't significant, therefore we decided to continue the model compression with XLNet. Moreover, it is noticeable that lower learning rates and low-medium train batches result in better accuracy scores (as we assumed).

Following the random search, we trained both models over the best hyperparameters found. The following graphs show the training loss vs validation loss for both models:

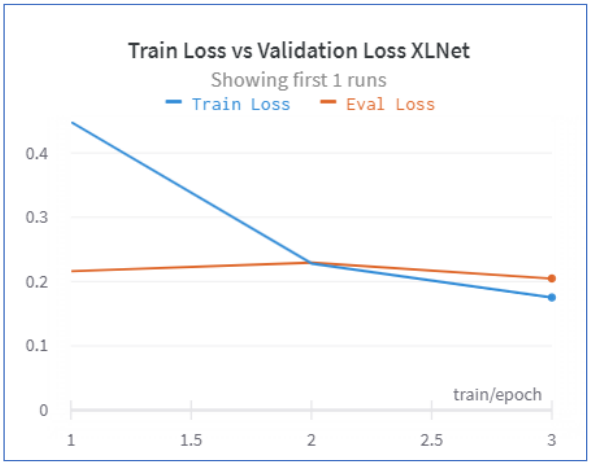

Figure 1.6 – Train vs Val Loss XLNet

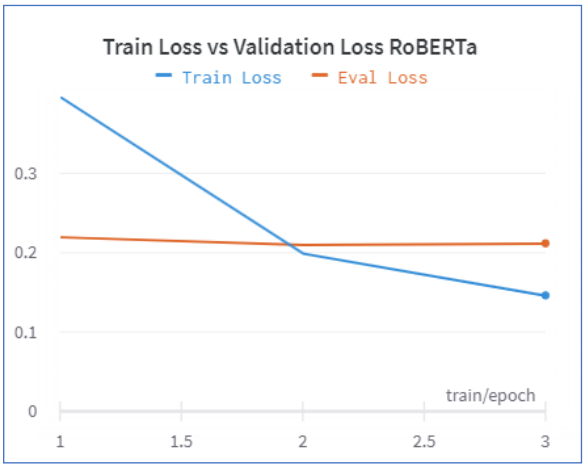

Figure 1.7 – Train vs Val Loss RoBERTa

As shown in the graphs, we haven't reached overfitting whereas no underfitting was reached either. XLNet's accuracy stands on 92.54%, while RoBERTa reached 92.7% accuracy.

Later, we have compressed the model using both techniques described in the methodology section above. Since we have already fine-tuned XLNet, we have trained the pruned model over 1 epoch (this way we avoided overfitting as well). While pruning has produced an accuracy score of 91.55% which is relatively close to the non-compressed model, applying the quantization method produced an accuracy score of 50%. However, by applying this method, we have been able to reduce the model's size from 469.32 MB to 299.45 MB (a decrease of 36%).

The table below describes each model's learning rate used for training, the number of epochs trained and the Accuracy and F1 metrics of the validation set (all of the models were trained on the same batch size of 17 reviews).

Model	Learning Rate	Epochs	Accuracy	F1
XLNet (Regular)	2.6785e-5	3	92.54%	92.57%
RoBERTa	1.1695e-5	3	**92.70%**	**92.72%**
XLNet Pruned	2.6785e-5	1	91.55%	91.31%
XLNet Quantized	-	-	50.00%	66.66%

Figure 1.8 – Tracked Metrics Comparison

3.2 Computational Power Usage

As the trained models' size is relatively big, and the dataset contained a large number of reviews (25K), a high computational power was required to perform an efficient random search, training, and compression of the models. Thus, we decided to run these processes on the computer's GPU. We have used NVIDIA's GeForce RTX 3080 GPU (10GB) and tried to maximize the available memory for training.

The graph below shows the GPU usage for the random search over both models:

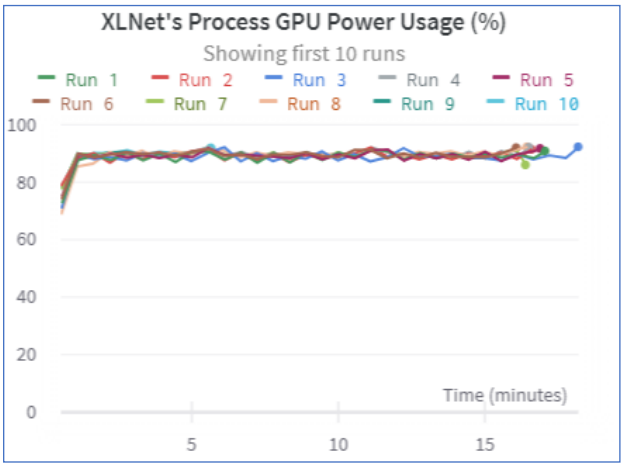

Figure 1.9 – XLNet's GPU Power Usage

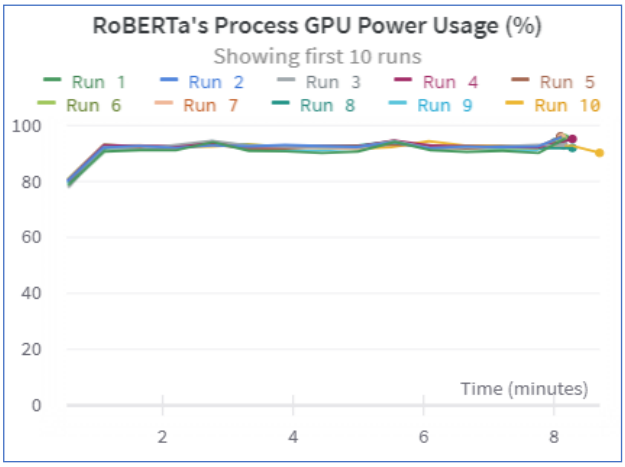

Figure 1.10 – RoBERTa's GPU Power Usage

As shown, we have managed to utilize around 90% of the GPU available for training (for every run in each model). XLNet's training time was overall higher than RoBERTa's training time (about double the training time).

3.3 Ablation Study

A neural network ablation study investigates the performance of an AI system by removing certain components to understand the contribution of the component to the overall system [8]. In our work, we have performed an ablation study by comparing the performance of the models with and without applying our pre-processing functions on the dataset to check if the pre-processing functions contribute to the model's performance. We have discovered that the accuracy of the models without applying the pre-processing functions was lower by 0.2% than the accuracy of the models after applying the pre-processing functions (a higher difference was obtained in specific random search runs).

4. Conclusions & Discussion

Sentiment Analysis tasks are among the utmost tasks in the Deep Learning area in general and in the NLP field specifically. It is an undeniable fact that human language is relatively complex to be understood by machines, which can produce correlation errors where negatively said words have a positive association and vice versa. Hence, a sentimental analysis of IMDB's movie reviews can be a challenging task. We received an un-cleaned dataset of reviews and applied several pre-processing techniques to clean it. We have also managed to provide a POC of these techniques by performing an ablation study. We have then chosen two pre-trained models (XLNet and RoBERTa) to assist with the task and performed a large random search over these models to find the best hyperparameters for fine-tuning the models. We have fine-tuned the models over the hyperparameters that were found to be the most effective and received a high accuracy score for both models. Although RoBERTa's performance was slightly better than XLNet's performance (92.70% compared to 92.54% accordingly), we have chosen to apply two compression methods (pruning and quantization) on the XLNet model as the pruning technique resulted in better accuracy for XLNet. We have pruned 50% of the weights in each feed-forward layer and managed to maintain a high accuracy score (91.50%). We have also managed to reduce the size of the model by 36% using quantization on the feed-forward layers, however the accuracy obtained after applying this method was 50% (like a random choice model), probably because we have set the scale of the quantization to 0 and caused a ripple effect on the

weights (a similar phenomenon to the Vanishing Gradient Effect). Nevertheless, we haven't searched over different hyperparameters we have set on the pre-processing stage (i.e. the pruning amount and the quantization scale), and over the tokenization maximum length (that could possibly be enlarged), both of which can deeply affect the models' performance. Even though the fine-tuning we have performed resulted in high accuracy, we advise to perform more research over the vast amount of possible hyperparameters combinations in each of the pre-processing and training stages.

5. References

[1] Shaukat, Z., Zulfiqar, A.A., Xiao, C. et al. Sentiment analysis on IMDB using lexicon and neural networks. SN Appl. Sci. 2, 148 (2020).

[2] Kannan, S., Gurusamy, V., Vijayarani, S., Ilamathi, J., Nithya, M., Kannan, S., & Gurusamy, V. (2014). Preprocessing techniques for text mining. International Journal of Computer Science & Communication Networks, 5(1), 7-16.

[3] K. Pipalia, R. Bhadja and M. Shukla, "Comparative Analysis of Different Transformer Based Architectures Used in Sentiment Analysis," 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART), 2020, pp. 411-415, doi: 10.1109/SMART50582.2020.9337081.

[4] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of machine learning research, 13(2).

[5] Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019, July). Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 2623-2631).

[6] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

[7] Buciluǎ, C., Caruana, R., & Niculescu-Mizil, A. (2006, August). Model compression. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 535-541).

[8] Meyes, R., Lu, M., de Puiseau, C. W., & Meisen, T. (2019). Ablation studies in artificial neural networks. arXiv preprint arXiv:1901.08644