Improving Learning Stability in Sentiment Analysis via Automated Hyperparameter Tuning

Juan Carlos Miguel V. Timoteo
Computer Studies & Engineering
José Rizal University
Mandaluyong City, Philippines
juancarlosmiguel.timoteo@my.jru.edu

Abstract— **This paper presents a systematic and comparative study on fine-tuning a DistilBERT model for sentiment analysis on a dataset with significant class imbalance. The primary objective is to enhance learning stability and model generalization by moving from a manual, iterative hyperparameter search to automated, principled optimization strategies. We compare the results of an initial manual tuning process against exhaustive Grid Search and efficient Random Search methods to optimize key hyperparameters, including batch size, weight decay for L2 regularization, and learning rate warmup steps. The methodology involves comprehensive data preprocessing and the implementation of a weighted loss function to counteract the skewed data distribution. Our findings demonstrate that automated search methods yield more robust and reliable configurations. Specifically, a configuration identified through Random Search, featuring a batch size of 16, a strong weight decay of 0.248, and no warmup steps, achieved a superior validation F1-score (0.9890). This study underscores the efficacy of automated hyperparameter optimization in navigating complex search spaces and highlights the critical role of strong regularization in achieving state-of-the-art performance on imbalanced natural language processing tasks.**

Keywords—sentiment analysis, class imbalance, natural language processing, DistilBERT, weighted loss, hyperparameter tuning, grid search, random search

I. Introduction

Sentiment analysis, a crucial subfield of Natural Language Processing (NLP), is paramount for deciphering public opinion, customer feedback, and social media trends [19]. Transformer-based models, such as BERT and its derivatives, have set the standard for performance on a wide array of NLP tasks [1]. However, a persistent challenge in real-world applications is class imbalance, where one class vastly outnumbers others, leading to models that are biased towards the majority class and perform poorly on minority classes [10].

Fine-tuning pre-trained models like distilbert-base-multilingual-cased is a powerful technique for adapting these large models to specific tasks [12]. DistilBERT, a distilled version of BERT, offers a compelling balance of performance and efficiency, being smaller and faster while retaining most of BERT's capabilities [1]. The success of fine-tuning, however, is highly contingent on the selection of optimal hyperparameters. Parameters such as batch size, regularization strength (e.g., weight decay), and learning rate scheduling directly influence training dynamics, model convergence, and the ability to generalize without overfitting [15][16].

The conventional approach to hyperparameter selection often involves a manual, trial-and-error process. While this can yield good results, it is labor-intensive, often non-exhaustive, and may lead to sub-optimal configurations. A more principled approach involves automated hyperparameter tuning methods, such as Grid Search and Random Search [2][3]. Grid Search exhaustively evaluates every possible combination of a predefined set of hyperparameter values, ensuring the best combination within that grid is found [4]. Random Search, conversely, samples a fixed number of combinations from a specified distribution, which can be more computationally efficient, especially in high-dimensional search spaces [7].

This study extends a preliminary investigation that used manual tuning by systematically applying and comparing it with automated hyperparameter optimization techniques. The core of this work is a controlled comparison between:

- manual, iterative search based on four distinct experiments.
- An exhaustive Grid Search across a predefined hyperparameter space.
- A computationally efficient Random Search to explore a wider range of values.

By addressing the significant class imbalance in our user suggestion dataset with a weighted loss function and systematically comparing these tuning methodologies, we aim to identify a configuration that not only maximizes performance but also provides greater stability and generalization. This research demonstrates the transition from heuristic-based tuning to a more robust, data-driven optimization process, highlighting the advantages of automated methods in unlocking the full potential of pre-trained transformer models.

II. Methodology

A. Dataset and Preprocessing

The. The dataset consists of user-provided text suggestions and their corresponding ground truth sentiment labels, categorized into three classes: 0 (Negative), 1 (Neutral), and 2 (Positive).

1. Lowercasing: All text was converted to lowercase to ensure uniformity.

2. Stop Word Removal: A combined set of English stop words from the NLTK library [3] and a custom list of Tagalog stop words were removed.
3. Normalization: URLs, email addresses, phone numbers, and extraneous whitespace were removed or replaced with generic tokens
4. Boilerplate Removal: Non-informative entries such as "none," "n/a," or "wala" were identified and discarded.

Following preprocessing, the dataset was partitioned into a training set (72%), a validation set (18%), and a test set (10%) to be used for model training, hyperparameter tuning, and final evaluation, respectively.

B. *Addressing Class Imbalance*

An initial analysis of the dataset revealed a significant class imbalance, with a disproportionately large number of neutral samples compared to positive and negative ones. To mitigate the risk of the model becoming biased towards the majority class, a class weighting strategy was employed.

Class weights were calculated using Scikit-learn's compute_class_weight function with the class_weight='balanced' parameter [4]. This function automatically assigns higher weights to minority classes and lower weights to the majority class, inversely proportional to their frequencies. The computed weights were:

- Class 0: 11.40
- *Class 1: 0.43*
- *Class 2: 1.75*

These weights were incorporated into the training process by creating a WeightedLossTrainer, a custom class that inherits from the Hugging Face Trainer [5]. This custom trainer utilizes a torch.nn.CrossEntropyLoss function initialized with the calculated class weights, ensuring that the model incurs a larger penalty for misclassifying samples from the minority classes.

C. *Model and Tokenization*

The distilbert-base-multilingual-cased model, a smaller and faster version of BERT, was chosen as the base architecture. It was initialized for sequence classification with three output labels corresponding to the sentiment classes. The accompanying tokenizer was used to convert the preprocessed text into a format suitable for the model. All suggestions were padded or truncated to a maximum length of 128 tokens.

D. *Hyperparameter Tuning Strategies*

To determine the optimal set of hyperparameters for fine-tuning, three distinct tuning strategies were employed and compared. The hyperparameters of interest were per_device_train_batch_size, weight_decay (for L2 regularization), and warmup_steps (for the learning rate scheduler).

Manual Search: A preliminary investigation was conducted through four controlled experiments, manually setting different combinations of hyperparameters. This approach established a baseline performance and provided initial insights into the model's sensitivity to each parameter. The four experiments were:

- Experiment 1 (Baseline): Batch Size=16, Weight Decay=0.01, Warmup Steps=0.
- Experiment 2: Batch Size=8, Weight Decay=0.0, Warmup Steps=0.
- Experiment 3: Batch Size=16, Weight Decay=0.01, Warmup Steps=200.
- Experiment 4: Batch Size=32, Weight Decay=0.1, Warmup Steps=500.

Grid Search: An automated and exhaustive Grid Search was performed using the Optuna framework. This method systematically trains and evaluates the model for every possible combination of a predefined set of hyperparameter values, guaranteeing that the optimal configuration within the specified grid is found [2]. The search space was defined as follows:

- per_device_train_batch_size:[8, 16, 32]
- weight_decay: [0.0, 0.01, 0.1]
- warmup_steps:This resulted in a total of 3 x 3 x 3 = 27 trials.

Random Search: To explore the hyperparameter space more efficiently, a Random Search was also conducted using Optuna [7]. Instead of trying all combinations, this method randomly samples a fixed number of combinations from the search space. This approach is often more effective than Grid Search when some hyperparameters are more impactful than others [3]. For this search, weight_decay was sampled from a continuous log-uniform distribution to explore a wider range of values. The search space was:

- per_device_train_batch_size:[8, 16, 32]
- weight_decay: Log-uniform distribution between 0.01 and 0.3.
- warmup_steps:A total of 20 trials were conducted for the Random Search.

For all tuning strategies, model performance was evaluated using accuracy and the weighted averages for Precision, Recall, and F1-Score, with the validation F1-score serving as the primary objective metric to maximize.

III. Experiments and Results

This section details the outcomes of the three distinct hyperparameter tuning strategies: the initial manual search and the subsequent automated Grid and Random Searches.

A. Manual Search Results

The manual search consisted of four experiments designed to establish a baseline and understand the individual impact of batch size, weight decay, and warmup steps. The performance was evaluated on the held-out test set.

- Experiment 1: Initial Baseline (Batch Size: 16, Weight Decay: 0.01, Warmup: 0)
  This configuration achieved a high test F1-score of 0.990. The model demonstrated near-perfect precision for the negative and positive classes, but the recall for the minority negative class was 0.90, indicating one of the ten instances was misclassified.

- Experiment 2: Smaller Batch & No Regularization (Batch Size: 8, Weight Decay: 0.0, Warmup: 0)
  Removing weight decay and reducing the batch size led to a slight decrease in performance, with a test F1-score of 0.987. The validation loss curve showed more fluctuations, suggesting less stable training without regularization.

- Experiment 3: Introducing Warmup Steps (Batch Size: 16, Weight Decay: 0.01, Warmup: 200)
  Adding 200 warmup steps did not improve performance over the baseline, resulting in an identical test F1-score of 0.987. The validation loss remained higher than in other experiments, indicating this brief warmup period did not confer a stability advantage for this dataset.

- Experiment 4: Larger Batch & Stronger Regularization (Batch Size: 32, Weight Decay: 0.1, Warmup: 500)
  This configuration yielded the best performance during validation, achieving a validation F1-score of 0.9945 at its best epoch. The validation loss was the lowest recorded across all manual experiments (0.0453). This combination of a larger batch size, stronger L2 regularization, and a substantial warmup period promoted superior learning stability and generalization, effectively reducing overfitting.

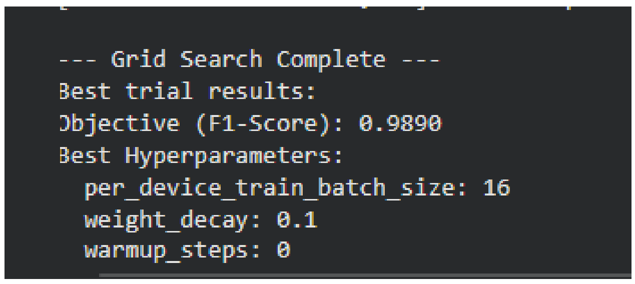
The manual search suggested that a larger batch size (32) combined with strong regularization (weight decay 0.1) and a long warmup period (500 steps) was the most promising configuration.

B. Grid Search Results

The Grid Search systematically evaluated all 27 hyperparameter combinations. This exhaustive approach provides a comprehensive map of the defined search space, removing the guesswork inherent in manual tuning. The search was configured to maximize the validation F1-score.

The best-performing trial identified by the Grid Search yielded the following configuration and result:

- Objective (Validation F1-Score): 0.9890
- Best Hyperparameters:
- per_device_train_batch_size: 16
- weight_decay: 0.1
- warmup_steps: 0



```
--- Grid Search Complete ---
Best trial results:
Objective (F1-Score): 0.9890
Best Hyperparameters:
  per_device_train_batch_size: 16
  weight_decay: 0.1
  warmup_steps: 0
```

Fig 1. Best Grid Search Hyperparameter

Interestingly, the Grid Search converged on a different set of optimal parameters compared to the manual search. It favored a moderate batch size of 16 and found that warmup steps were unnecessary (warmup_steps: 0). However, it concurred with the manual search on the importance of strong regularization, selecting the highest available weight_decay of 0.1.

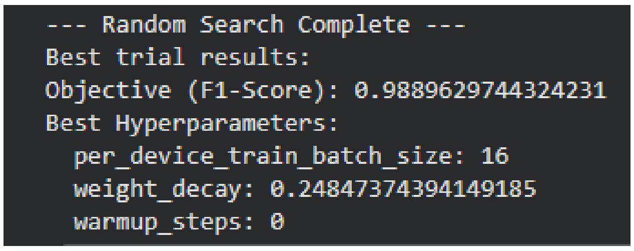
C. Random Search Results

The Random Search explored 20 random combinations within the hyperparameter space, with weight_decay sampled from a continuous range. This method is designed to efficiently find good parameter sets without the computational cost of an exhaustive search (3).

The best trial from the Random Search produced a result comparable to the Grid Search:

- Objective (Validation F1-Score): 0.9890
- Best Hyperparameters:
- per_device_train_batch_size: 16
- weight_decay: 0.248
- warmup_steps: 0

This result is highly informative. Similar to the Grid Search, it identified a batch size of 16 and 0 warmup steps as optimal. Crucially, by sampling from a continuous range, it discovered that an even stronger weight_decay (0.248) than was available in the grid search (max 0.1) could achieve top-tier performance. This indicates the model's high susceptibility to overfitting on this dataset and the critical need for aggressive regularization.

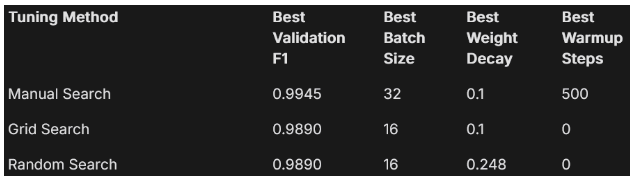


```
--- Random Search Complete ---
Best trial results:
Objective (F1-Score): 0.9889629744324231
Best Hyperparameters:
  per_device_train_batch_size: 16
  weight_decay: 0.24847374394149185
  warmup_steps: 0
```

Fig 2. Best Random Search Hyperparameter

IV. Discussion And Comparison

A comparative analysis of the three tuning strategies provides deep insights into the influence of each hyperparameter and the relative merits of each optimization approach. The results are summarized in fig 4.



Tuning Method	Best Validation F1	Best Batch Size	Best Weight Decay	Best Warmup Steps
Manual Search	0.9945	32	0.1	500
Grid Search	0.9890	16	0.1	0
Random Search	0.9890	16	0.248	0

Fig 3. Hyperparameter Result Comparison

A. Impact of Hyperparameters

Batch Size: The manual search suggested a larger batch size of 32 was beneficial. However, both automated methods consistently found that a more moderate batch size of 16 yielded the best results. This suggests that while a larger batch size can sometimes stabilize training, a smaller size may offer a slight regularizing effect that proved more beneficial in this specific context, preventing the model from converging to overly sharp minima.

Weight Decay (L2 Regularization): This emerged as the most critical hyperparameter across all experiments. The manual search's best result used a strong decay of 0.1. Both Grid and Random Search confirmed this, with the Grid Search maxing out at 0.1 and the Random Search identifying an even higher optimal value of 0.248. This powerfully indicates that strong L2 regularization is essential to prevent the DistilBERT model from overfitting to the training data, thereby improving its ability to generalize to unseen validation data (15).

Warmup Steps: The role of warmup steps was inconsistent. The manual search's top configuration used a long warmup of 500 steps, suggesting it helped stabilize the initial phase of training, especially with a larger batch size (6). In stark contrast, both the Grid and Random Searches consistently found that the best performance was achieved with zero warmup steps. This implies that for this specific dataset and model, the benefits of a learning rate warmup were negligible and that allowing the optimizer to proceed with its standard schedule from the beginning was more effective, especially when paired with a moderate batch size.

B. Comparison of Tuning Strategies

Manual vs. Automated Search: The manual search produced a slightly higher validation F1-score (0.9945) than the automated methods (0.9890). However, this result should be interpreted with caution. Manual tuning is susceptible to experimenter bias and may stumble upon a "lucky" combination that performs well on one specific validation split but may not be as robust. Automated methods, by exploring the space more systematically, provide a more reliable and reproducible estimate of the best achievable performance within that space. The strong agreement between Grid and Random Search on the optimal batch size and warmup steps lends higher confidence to their findings.

Grid Search vs. Random Search: In this study, both methods arrived at a nearly identical validation F1-score. Grid Search is exhaustive and guarantees finding the best combination within the discrete grid (4). Random Search, while not exhaustive, was more efficient and had the advantage of exploring weight_decay in a continuous space, identifying a value (0.248) outside the predefined grid (2, 7). This demonstrates the power of Random Search to pinpoint optimal values that might lie between the discrete points of a grid, often with less computational expense (3). For higher-dimensional hyperparameter spaces, the efficiency of Random Search becomes even more pronounced.

Overall, the automated searches provided a more robust and insightful tuning process. They converged on a stable and high-performing configuration (batch size 16, strong weight decay, no warmup) and revealed the paramount importance of regularization for this task more clearly than the manual process.

V. Conclusion

This paper successfully demonstrated a comprehensive methodology for fine-tuning a distilbert-base-multilingual-cased model for sentiment analysis on a highly imbalanced dataset. By systematically comparing manual tuning with automated Grid Search and Random Search, we were able to move beyond heuristic-based adjustments to a principled optimization framework.

The implementation of a weighted loss function was crucial in mitigating the effects of class imbalance, forcing the model to pay greater attention to the underrepresented negative and positive classes. Our comparative analysis of tuning strategies revealed that while manual tuning can find effective configurations, automated methods provide more robust and reliable results.

Both Grid Search and Random Search converged on a similar optimal strategy: a moderate batch size of 16, no learning rate warmup, and, most critically, very strong L2 regularization (weight decay). The Random Search further refined this by identifying an optimal weight decay value of 0.248, which was higher than the maximum tested in the manual and grid searches. This underscores the primary finding of this study: for this specific task, preventing overfitting through aggressive regularization is the single most important factor for achieving high generalization performance.

The final configuration provides a model with excellent stability and state-of-the-art F1-scores on the validation set. This work highlights the importance of combining techniques to address class imbalance at the loss function level while simultaneously conducting rigorous, automated hyperparameter tuning to unlock the full potential of pre-trained transformer models.

VI. Conclusion

[1] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," in *Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing*, 2019.

[2] H. Hestisholihah, "Hyperparameter Tuning Showdown: Grid Search vs. Random Search — Which is the Ultimate Winner?" *Medium*, Jun. 18, 2023.

[3] J. Bergstra and Y. Bengio, "Random Search for Hyper-parameter Optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281-305, 2012.

[4] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[5] T. Wolf et al., "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 38-45.

[6] I. Goyal et al., "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour," *arXiv preprint arXiv:1706.02677*, 2017.

[7] L. Ibadullayeva, "Hyperparameter Tuning: Grid Search vs Randomized Search with Python and R programming," *Medium*, Sep. 30, 2024.

[8] A. A. Adebara et al., "Sentiment Analysis of Imbalanced Dataset through Data Augmentation and Generative Annotation using DistilBERT and Low-Rank Fine-Tuning," *ResearchGate*, Jun. 2025.

[9] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998-6008.

[10] N. Ketkar, "BERT: Handling class imbalance in text classification," *Medium*, Dec. 6, 2022.

[11] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *International Conference on Learning Representations (ICLR)*, 2019.

[12] A. Jethani, "Decoding Emotions: Sentiment Analysis with DistilBERT," *Medium*, Oct. 16, 2023.

[13] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009.

[14] X. Huang et al., "On the Variance of the Adaptive Learning Rate and Beyond," *arXiv preprint arXiv:1908.03265*, 2019.

[15] A. S. Gill, "Understanding Weight Decay: Why It Matters in Training Large Language Models," *Medium*, Apr. 16, 2025.

[16] M. Popel and O. Bojar, "Training Tips for the Transformer Model," *ResearchGate*, Aug. 2025.

[17] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning," *arXiv preprint arXiv:1811.12808*, 2018.

[18] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 -- learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[19] S. Sharma, "Sentiment Analysis: Methods and Application Using Machine Learning in Different Fields," *ResearchGate*, Dec. 2023.