

Using Large Language Models for Data Augmentation in Text Classification Models

BOHDAN PAVLYSHENKO¹, MYKOLA STASIUK²

¹System Design Department, Ivan Franko National University of Lviv, Lviv, 79005 Ukraine (e-mail: bohdan.pavlyshenko@lnu.edu.ua)

²System Design Department, Ivan Franko National University of Lviv, Lviv, 79005 Ukraine (e-mail: mykola.stasiuk@lnu.edu.ua)

Corresponding author: M. Stasiuk (e-mail: mykola.stasiuk@lnu.edu.ua).

ABSTRACT This research considers the impact of data augmentation on multi-class text classification. A diverse news dataset comprising four categories was utilized for training and evaluation. Various transformer models, including BERT, DistilBERT, ALBERT, and RoBERTa, were employed to classify text across multiple categories. Based on the previous research on data augmentation, synonym replacement, antonym replacement, contextual word embedding, and the lambda method for data augmentation were chosen. Three mainstream LLMs were selected to investigate the capabilities of LLMs: LLaMA 3, GPT-4, and MistralAI. These models represent a diverse range of architectures and training data, allowing to assess the impact of different LLM capabilities on data augmentation performance. The performance of the aforementioned transformer models was evaluated using metrics such as accuracy, recall, precision, F1-score, training time, validation, and training loss. Experiments revealed that data augmentation significantly improved the performance of transformer models in text classification tasks, with lambda augmentation consistently outperforming other methods. However, model architecture and hyperparameter tuning also played a crucial role in achieving optimal results. RoBERTa, in particular, required careful hyperparameter adjustment to reach competitive performance levels. Obtained results have practical implications for developing NLP applications in low-resource languages, as data augmentation can help address the limitations of small datasets.

KEYWORDS augmentation; multi-class text classification; large language models; transformers; BERT; ALBERT; DistilBERT; XLM-RoBERTa;

I. INTRODUCTION

Transformer models have emerged as the dominant architecture in contemporary Natural Language Processing (NLP) [1]–[4]. Their capacity for parallel processing through self-attention mechanisms enables efficient handling of long-range dependencies and variable-length sequences, surpassing the limitations of previous models. While transformers excel in capturing intricate linguistic patterns, they are computationally demanding and require substantial amounts of data to achieve optimal performance. Moreover, they are susceptible to overfitting. Prominent examples of transformer architectures include BERT [5], RoBERTa [6], DistilBERT [7], and ALBERT [8].

Despite these challenges, the remarkable capabilities of

transformers are evident in the development of Large Language Models (LLMs), which have revolutionized the field. These models are built upon the transformer architecture and trained on massive datasets, enabling them to generate human-quality text, translate languages, write different kinds of creative content, and answer any questions in an informative way [9]–[12]. LLMs excel in capturing complex language patterns and dependencies due to their self-attention mechanism. However, their development is computationally intensive, requiring substantial resources. Additionally, they often exhibit biases in the training data, necessitating careful consideration during development and deployment. Despite these challenges, LLMs have become indispensable tools for various applications, from chatbots and virtual assis-

tants to content creation and language translation [13]–[17]. To achieve truly context-aware functionality within those applications, especially when those applications are used within smart cities, LLMs require access to vast amounts of real-time data. E. g. in IoT cities, the data is generated by sensors, cameras, and other connected devices. The efficient processing of this data is crucial for providing the necessary input to these LLMs [18].

Data augmentation is an important technique applied to enhance the performance of machine learning models, particularly in data-scarce scenarios [19]–[21]. Data augmentation mitigates overfitting and improves model generalization by artificially expanding the training dataset through various transformations. In natural language processing, augmentations can be applied at the character, word, or sentence levels, including techniques like synonym replacement, back translation, and noise injection [22], [23].

The advent of LLMs has introduced new possibilities for data augmentation. These powerful models, trained on massive amounts of text data, deeply understand language patterns and can generate highly realistic text samples. However, a question arises: to what extent can LLMs effectively contribute to data augmentation? While LLMs offer the potential to create diverse and complex augmented data, their generated samples might not permanently preserve the original data distribution or semantic meaning, potentially leading to performance degradation.

While recent studies have explored the potential of LLMs for data augmentation [24]–[27], their effectiveness compared to more traditional methods remains a subject of ongoing research. While LLMs have shown promise in generating diverse and realistic synthetic data, their ability to accurately capture the underlying distribution of the original data and improve model performance in various tasks still requires further investigation. Addressing challenges such as bias, quality control, and computational efficiency will be crucial in determining the practical applicability of LLMs for data augmentation in real-world scenarios.

This paper evaluates the impact of various augmentation techniques on the effectiveness and training efficiency of different transformer models applied to multi-class text classification tasks. Additionally, the study compares the performance of LLMs in generating augmented data with traditional augmentation methods.

II. MATERIAL AND METHODS

A subset of 1000 records from the dataset [28] for our experiments was utilized.

Table 1. Labels distribution.

Dataset	Label, quantity			
	1	2	3	4
Test	988	1004	1008	1000
Train	988	990	1006	1016

Table 1 shows the distribution of labels in the used dataset. As the table demonstrates, labels are almost evenly

distributed.

Augmented datasets were constructed by applying augmentation techniques to the original dataset, resulting in a threefold. In this research, word-level and sentence-level augmentations, as part of the non-LLM component, were utilized. These augmentations were done with the `nlpaug` library [29].

Building upon our previous research [30], which explored a comprehensive range of augmentation techniques, the focus was set on the ones listed below for this study due to their demonstrated effectiveness in enhancing model performance and addressing overfitting:

- synonym augmenter: replaces words with semantically similar alternatives;
- antonym augmenter: replaces words with semantically opposite terms;
- contextual word embeddings augmenter: leverages contextual word embeddings to identify and substitute top semantically similar words;
- lambada augmenter [31]: modifies textual input through an abstractive summarization process based on the lambada method.

Augmentation experiments were conducted on a system equipped with an Intel i7-13700K CPU and NVIDIA GeForce GTX 4070 Ti GPU.

In addition to traditional augmentation methods, three LLMs were evaluated: LLaMA 3 [32], MistralAI [33], and [34].

The following message was a prompt for LLMs: ‘You will rephrase the text, that is given as input’. The GPT-4 model was accessed through the OpenAI API. The elapsed time between the API request and the response was used to calculate the execution time. The LLaMa 3 and MistralAI models were accessed through Google Colab. The last two models were run on the instance with an NVIDIA A100 GPU, and the model’s response generation time was considered the execution time. The LLaMa 3 and MistralAI models were loaded from the HuggingFace portal and are accessible under the names ‘meta-llama/Meta-Llama-3-8B-Instruct’ and ‘mistralai/Mistral-7B-Instruct-v0.3’, respectively.

Transformer models were trained on the NVIDIA GeForce GTX 4070 Ti GPU. All models and tokenizers used in this study are publicly available on Hugging Face, accessible by their respective names: `bert-base-uncased`, `distilbert-base-uncased`, `xlm-roberta-base`, and `albert-base-v2`.

The `DataColator` class from the Hugging Face Transformers library was used for data batching. The `Trainer` and `TrainingArguments` classes were utilized to streamline the training process. All experiments were conducted within the PyTorch framework, leveraging the capabilities of the Transformers package. Models were initially trained for five epochs. Model evaluation and saving were performed after each epoch. The weight decay for every experiment was 0.01.

To investigate the influence of data augmentation, the aforementioned transformer models were trained on eight datasets: the original dataset and seven augmented versions created using the previously described methods. The performance of each model was evaluated using a comprehensive set of metrics, including validation and training loss, accuracy, precision, F1-score, and recall.

III. RESULTS

Table 2. Augmentation Processing Time

Augmentation	Dataset	
	Test	Train
Time, s.ms		
Word-level augmentation		
Synonym	3.91	4.50
Antonym	4.82	4.61
Embeddings	154.59	209.02
Sentence-level augmentation		
Lambda	136.13	123.31
LLMs		
GPT-4	7770.61	7794.62
LLaMa 3	11070.71	11261.63
Mistralai	11453.31	11299.63

Table 2 shows the time consumed by different augmentation techniques while modifying the original dataset. Augmentations can be categorized into groups based on their execution times. Synonym and antonym augmentations were the fastest, completing in less than 5 seconds. In contrast, contextual word embeddings and the lambda method took approximately 2 minutes each.

Augmentations using LLMs were the most time-consuming, requiring over 2 hours and 9 minutes per dataset. GPT was the fastest LLM, with an approximately one-hour runtime shorter than LLaMa. The Mistralai took the longest, with execution times of around 3 hours and 10 minutes.

Table 3. Models training time.

Augmentations	Transformer model			
	BERT	DistilBERT	RoBERTa	ALBERT
Time, s.ms				
No augmentation	69.2835	44.6286	164.9856	33.4917
nlpaug				
Synonym	239.6693	150.0055	501.0972	119.4692
Antonym	238.9841	153.0150	500.0100	118.2244
Embeddings	237.5268	147.8533	498.3375	116.6700
Lambda	245.2489	151.2111	510.4234	133.0340
LLM				
GPT-4	244.5692	150.5958	506.1688	127.0171
LLaMa 3	242.1237	149.4536	502.1718	123.2351
MistralAI	246.1742	151.1493	509.1011	129.1767

Table 3 presents the training time required for various transformer models. ALBERT exhibited the shortest training time, followed by DistilBERT. The original BERT model took moderate time, while RoBERTa required the most prolonged training.

A consistent pattern emerged among BERT, DistilBERT, and ALBERT regarding training time based on data augmentation techniques. Data augmented with the Lambda

method resulted in the longest training times among all models trained on nlpaug-modified datasets. Conversely, contextual word embedding augmentation led to the shortest training times for these models. Antonym augmentation yielded moderately long training times, while synonym augmentation was time-consuming but less so than the lambda method.

Models trained on LLM-augmented datasets consistently required longer training times than those trained on nlpaug-modified datasets. Among LLM-augmented models, the LLaMa-augmented model trained the fastest, followed by the GPT-4 augmented model. The MistralAI LLM-augmented model had the longest training time.

DistilBERT's behavior aligned with this pattern, with a notable exception: antonym augmentation resulted in the longest training time for this model, while embedding augmentation remained the fastest. Other DistilBERT models aligned with the overall training time trends observed in the other transformer models.

The first experiment involved training each model on every dataset using the following hyperparameters: a learning rate of $1e^{-5}$ and training and evaluation batches per device set to 8. After training each model, results showed that none effectively addressed overfitting. Moreover, accuracy, precision, F1-score, and recall were lower than those of models trained on the original dataset.

Due to these challenges, a second experiment involving hyperparameter adjustments was conducted:

- decreased the learning rate of the models to $1e^{-6}$;
- decreased training and evaluation batches per device to 4.

Retraining the models with these modified hyperparameters yielded improved results.

The following pages present images that contain training and evaluation metrics for four models: BERT, DistilBERT, ALBERT, and RoBERTa. Every model's metrics are organized in a 2x3 grid format. The top row of each grid displays training and validation loss curves, providing insights into the model's learning progress and potential overfitting. The middle row showcases precision and F1-score, reflecting the model's ability to identify positive instances correctly. The bottom row illustrates accuracy and recall, indicating the overall correctness and completeness of the model's predictions. A legend accompanying each image clarifies the correspondence between the plots and the specific augmentations applied to the dataset for training the model.

Figure 1 presents the training and evaluation results for BERT models using different datasets. For the model trained on the original dataset, training loss initially increases after the first epoch before decreasing. Meanwhile, evaluation loss steadily declines. Training and evaluation losses decreased for other models, indicating no overfitting.

Figure 1 also presents evaluation metrics for the trained models. The model trained on the original, unaugmented dataset achieved metrics around 80%. The model trained on

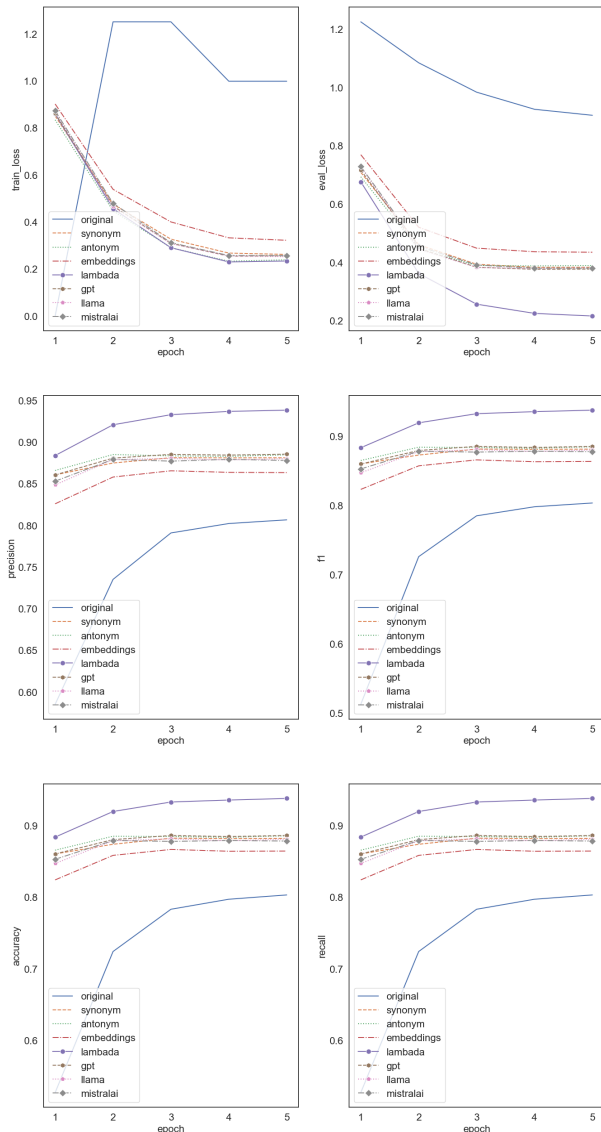


Figure 1. Bert models' training result.

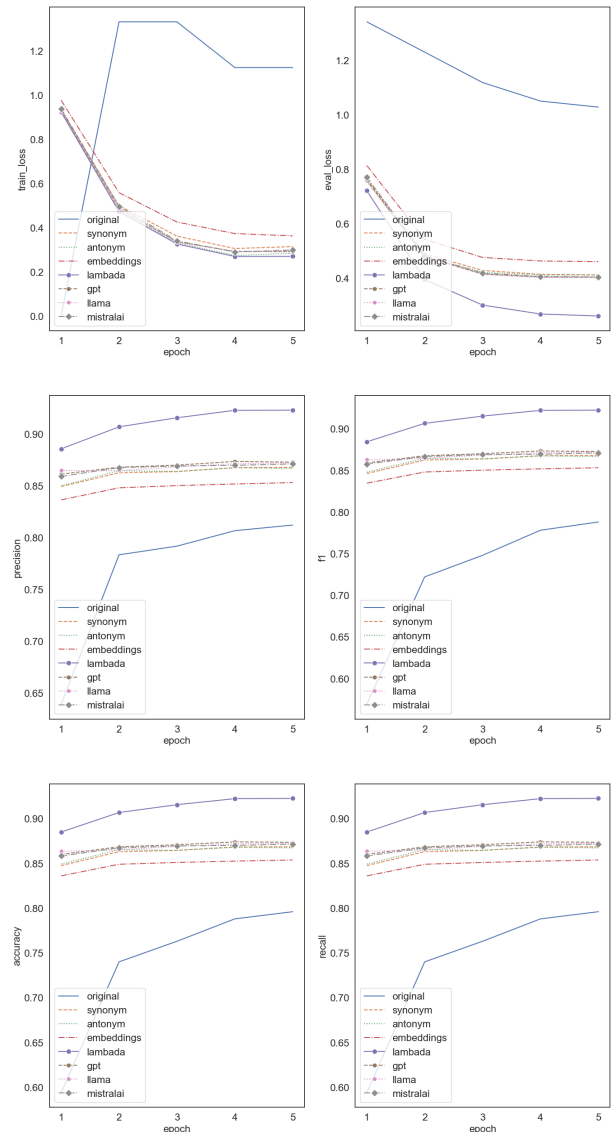


Figure 2. DistilBERT models' training result.

the dataset augmented with the Lambada method demonstrated the best performance with metrics near 93%. Other models trained on augmented datasets generally improved results to around 88%, with the model trained on GPT-augmented data performing best in this group, followed by those trained on antonym, synonym, and LLaMA augmented data. The model trained on MistralAI-augmented data scored 87%, while the model trained on contextual word embeddings augmentation achieved the lowest metrics at 86% among the augmented models.

Figure 2 demonstrates the training results for the DistilBERT models. Like the BERT models, all models trained on augmented datasets did not exhibit overfitting, unlike those trained on the original, unaugmented dataset.

At the same time, Figure 2 shows evaluation metrics for DistilBERT models. Similar to BERT, the model trained on the original dataset achieved lower metrics, around 79%.

The lambada-augmented model again demonstrated the best performance with metrics near 92%, followed by the GPT-augmented model at 87%. Unlike BERT, the LLaMA and MistralAI-augmented models also achieved scores of around 87%. Models trained on synonym and antonym augmented data obtained metrics around 86%. As with BERT, the contextual word embeddings augmentation resulted in the lowest metrics at 85%.

Figure 3 presents the training and evaluation results for ALBERT models using different datasets. Unlike DistilBERT, all ALBERT models, including those trained on augmented datasets, exhibited signs of overfitting after two training epochs.

Evaluation metrics show that the models trained on the lambada, antonym, and contextual word embeddings augmented datasets maintained decreasing evaluation losses until the second epoch. However, all other models experienced

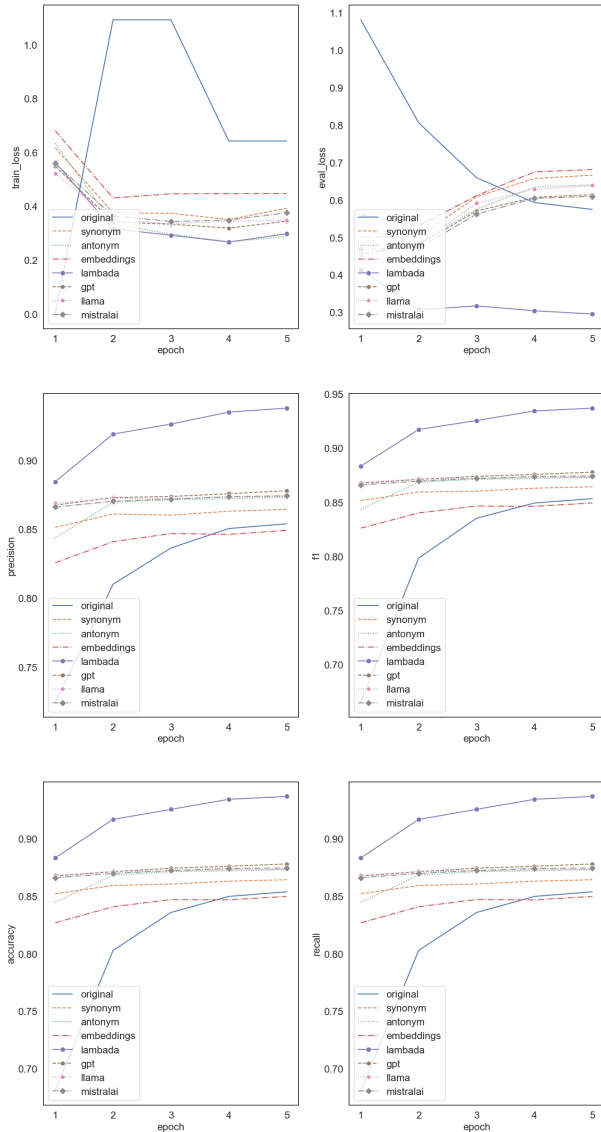


Figure 3. ALBERT models' training result.

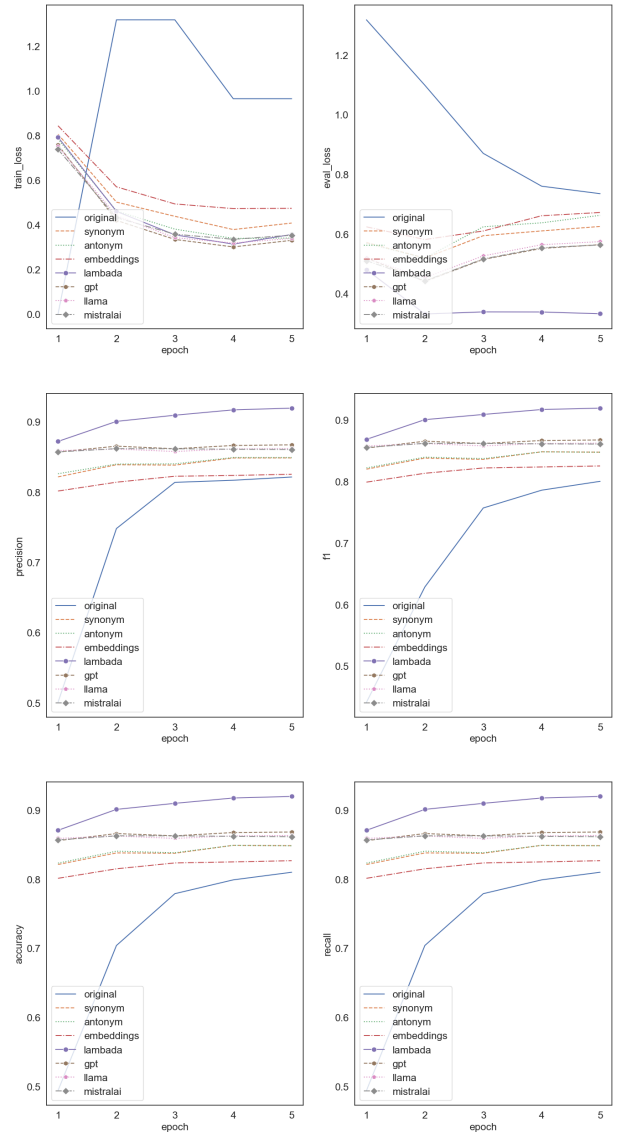


Figure 4. RoBERTa models' training result.

a rise in evaluation loss after this point.

Despite the overfitting, the ALBERT model trained on the lambada-augmented dataset still achieved the highest performance with a score of 93%. The models trained on the GPT-4, MistralAI, LLaMA-3, and antonym-augmented datasets all obtained scores around 87%. The model trained on the synonym-augmented dataset scored 86%, and the models trained on the original and contextual word embeddings augmented datasets both achieved 85%.

Figure 4 presents the training and evaluation results for RoBERTa models using different datasets. Unlike ALBERT, all RoBERTa models, including those trained on augmented datasets, maintained decreasing evaluation losses until the second training epoch. However, after this point, the evaluation loss for all models either remained stable or increased, indicating potential overfitting.

Regarding evaluation metrics, the RoBERTa model

trained on the lambada-augmented dataset achieved the highest performance with a score of 91%. The models trained on the GPT-4, LLaMA-3, and MistralAI-augmented datasets all obtained scores of around 86%. The models trained on the synonym and antonym-augmented datasets scored 84%, and the models trained on the original and contextual word embeddings augmented datasets both achieved 82% and 81%, respectively.

RoBERTa's suboptimal performance on the news classification task is partly attributed to overfitting, as evidenced by the increasing evaluation loss during training. Additional experiments with RoBERTa models were performed. The learning rate was adjusted to be $1e^{-5}$, and per device training and evaluation batch were decreased to 2. These changes allowed the model's efficiency to improve.

After this experiment, the next was found: only the lambada-augmented dataset allowed the model to overcome

overfitting. In contrast, all other augmented and original datasets exhibited signs of overfitting. The model trained on the lambada-augmented dataset achieved the highest performance with a score of 99%. Other augmented models, such as those trained on GPT (87%), antonym (86%), MistralAI (85%), synonym (87%), and LLaMA (85%) augmented datasets, also showed improvements but did not surpass the performance of the original, unaugmented model (87%). The model trained on contextual word embeddings augmentation performed slightly worse, achieving a score of 84%.

The performance differences observed in RoBERTa's evaluation metrics can be attributed to several factors. The Lambada dataset, explicitly designed for language modeling, might align more with RoBERTa's architecture and training objectives than other datasets like GPT, LLaMA, and MistralAI. Additionally, RoBERTa's inherent design, pre-training, and hyperparameter tuning could affect its performance on specific tasks.

IV. CONCLUSIONS

This research investigates the impact of data augmentation on multi-class text classification using transformer models. Various transformer models were utilized, including BERT, RoBERTa, DistilBERT, and ALBERT. To augment the dataset, traditional methods from the nlpaug library, including synonyms, antonyms, contextual word embeddings, and lambada, as well as advanced LLMs like GPT-4, LLaMA 3, and MistralAI, were employed.

The computational cost of augmentation techniques varies widely depending on their complexity. Synonym and antonym augmentation, which involve simple word substitutions, are the fastest, typically taking less than 5 seconds. While more involved than synonym/antonym replacement, contextual word embedding augmentation and the lambada method still complete relatively quickly in around 2.5 minutes. These techniques leverage pre-trained word embeddings to generate contextually relevant replacements. LLM-based augmentations, on the other hand, are the most complex and time-consuming. LLMs can generate entirely new text segments or rewrite existing text, often requiring over 2 hours to complete.

The analysis shows that augmentation techniques influence model training time. The time consumed by each model training with augmented datasets follows a consistent pattern, except for DistilBERT.

Across all models, lambada augmentation consistently outperformed other techniques, achieving the highest scores in most cases. For example, BERT and DistilBERT achieved F1-scores of 93% and 92%, respectively, with lambada augmentation, compared to around 80% for the original, unaugmented datasets. Augmentation techniques based on LLMs (GPT-4, LLaMA 3, and MistralAI) generally outperformed traditional methods. However, their effectiveness varied, suggesting that the choice of LLM and the specific augmentation strategy can influence results.

While Lambada consistently improved performance, the specific benefits varied across models. For instance, RoBERTa required adjusted hyperparameters to achieve enhanced results with lambada augmentation, highlighting the importance of careful tuning for individual models. The overall trend was that LLM-based augmentations generally outperformed traditional nlpaug methods, suggesting the potential benefits of leveraging large language models for data augmentation.

While data augmentation can effectively mitigate overfitting, its effectiveness varies across models. Factors such as model architecture, hyperparameters, and data characteristics influence a model's susceptibility to overfitting. For example, models with larger parameter sizes or complex architectures may be more prone to overfitting, while carefully tuned hyperparameters and well-chosen augmentation techniques can help prevent this issue. Additionally, datasets with imbalanced classes or limited diversity may be more susceptible to overfitting.

In the next step, we plan to investigate the augmentation capabilities of various transformer models and more straightforward approaches for other languages. While LLMs, such as GPT family models, MistralAI, and LLaMA, may also be considered, our primary focus will be on these more traditional techniques. The results of this research may contribute to developing more effective natural language processing tools for different languages and help bridge the language gap in the field.

References

- [1] N. Patwardhan, S. Marrone, and C. Sansone, "Transformers in the real world: A survey on nlp applications," *Information*, vol. 14, no. 4, p. 242, 2023. [Online]. Available: <https://doi.org/10.3390/info14040242>
- [2] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz et al., "Huggingface's transformers: State-of-the-art natural language processing," arXiv preprint arXiv:1910.03771, 2019. [Online]. Available: <https://doi.org/10.48550/arXiv.1910.03771>
- [3] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Q. Liu and D. Schlangen, Eds. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: <https://doi.org/10.18653/v1/2020.emnlp-demos.6>
- [4] A. M. Braşoveanu and R. Andonie, "Visualizing transformers for nlp: a brief survey," in *2020 24th international conference information visualisation (IV)*. IEEE, 2020, pp. 270–279. [Online]. Available: [10.1109/IV51561.2020.00051](https://doi.org/10.1109/IV51561.2020.00051)
- [5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: <https://doi.org/10.48550/arXiv.1810.04805>
- [6] Y. Liu et al., "Roberta: A robustly optimized bert pretraining approach," 2019. [Online]. Available: <https://doi.org/10.48550/arXiv.1907.11692>
- [7] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," 2020. [Online]. Available: <https://doi.org/10.48550/arXiv.1910.01108>
- [8] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," 2020. [Online]. Available: <https://doi.org/10.48550/arXiv.1909.11942>
- [9] W. X. Zhao et al., "A survey of large language models," 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2310.06825>

