International Journal of Internet, Broadcasting and Communication Vol.16 No.3 262-273 (2024)
http://dx.doi.org/10.7236/IJIBC.2024.16.3.262

IJIBC 24-3-28

Vulnerability Threat Classification Based on XLNET AND ST5-XXL model

Chae-Rim Hong[1], Jin-Keun Hong[2]*

[1]Graduate Student, Department of AI & Bigdata, aSSIST University, Korea
[2]Professor, Div. of Advanced IT, Baekseok University, Korea
E-mail: jump071383@stud.assist.ac.kr, jkhong@bu.ac.kr

Abstract

We provide a detailed analysis of the data processing and model training process for vulnerability classification using Transformer-based language models, especially sentence text-to-text transformers (ST5)-XXL and XLNet. The main purpose of this study is to compare the performance of the two models, identify the strengths and weaknesses of each, and determine the optimal learning rate to increase the efficiency and stability of model training. We performed data preprocessing, constructed and trained models, and evaluated performance based on data sets with various characteristics. We confirmed that the XLNet model showed excellent performance at learning rates of 1e-05 and 1e-04 and had a significantly lower loss value than the ST5-XXL model. This indicates that XLNet is more efficient for learning. Additionally, we confirmed in our study that learning rate has a significant impact on model performance. The results of the study highlight the usefulness of ST5-XXL and XLNet models in the task of classifying security vulnerabilities and highlight the importance of setting an appropriate learning rate. Future research should include more comprehensive analyzes using diverse data sets and additional models.

Keywords: *Machine Learning, Deep Learning, Feature, Model Training*

1. Introduction

In the field of artificial intelligence, the importance of models for analyzing and processing complex text data is continuously emphasized. Identifying and analyzing potential security vulnerabilities in source code is an important task in cybersecurity. Furthermore, research is needed on whether Transformer-based language models such as ST5-XXL and XLNet can show good performance in text classification and tasks in the cyber security area. In this paper, we intend to apply it to vulnerability classification using ST5-XXL and XLNet models. By comparing the performance of these two models, we aim to understand the characteristics of the model, determine the optimal learning rate, and determine the appropriateness of applying the model in vulnerability classification tasks. Optimizing the performance of Transformer-based models in vulnerability classification tasks requires a clear understanding of the relationship between learning rate and model performance. If the learning rate is too low or too high, the training process may be too slow or unstable, affecting the efficiency or accuracy of the model. Therefore, it is necessary to analyze how different learning rates affect the performance of ST5-XXL and XLNet models.

Manuscript Received: June. 25, 2024 / Revised: June. 30, 2024 / Accepted: July. 5, 2024
Corresponding Author: jkhong@bu.ac.kr
Tel: +82-41-550-2445
Professor, Faculty of Advanced IT, Baekseok, University, Korea

Therefore, in this paper, we train ST5-XXL and XLNet models at various learning rates and compare and analyze their performance to determine the optimal learning rate.

The research methods used in this paper are as follows. First, data is collected and preprocessed. Collect vulnerability-related datasets, verify the quality of the data, and correct it if necessary. Second, create and train a model. Set up ST5-XXL and XLNet models and train them at various learning rates. Third, evaluate performance. The data set is divided into training data, validation data, and test data, and the training data is trained according to a designated model and performance is evaluated. At this time, indicators of training loss and accuracy are used. Finally, analyze the results. By analyzing the impact of learning rate on model performance, we determine the optimal learning rate and visualize it. The structure of this paper is as follows. Chapter 2 introduces the experimental models ST5-XXL and XLNet, Chapter 3 discusses experiments and results, and Chapter 4 concludes the paper.

2. Experimental Models: ST5-XXL and XLNet

In the study of this paper, experimental data were obtained from.... Additionally, the models used in the experiment are ST5-XXL and XLNet models. The ST5-XXL model is a scalable sentence encoder of ST5 (Pre-trained Text-to-Text) and is a transformer-based language model that is excellent for identifying and analyzing potential security vulnerabilities in source code through text classification.

Chowdhery et al. trained a translator language model using an impressive 540 billion (540B) parameters [1]. Additionally, J. Ni and his researchers conducted research on extracting sentence representations from the T5 model [2]. The researchers explored optimal methods for deriving sentence embeddings from the T5 model and evaluated how well the original T5 sentence embeddings performed in downstream tasks such as classification and similarity measures. Their study confirmed the fundamental performance of the original T5 sentence embeddings and studied how contrasting sentence embeddings improve tasks such as natural language reasoning and question answering. Using datasets for natural language inference and question answering, performance improvement was evaluated based on semantic similarity between sentences.

The analysis focused on how expanding the number of parameters of the T5 model improves sentence embedding performance. The principles of self-attention, feedforward network, and location encoding were applied to this model. The transformer model uses the concept of positional encoding to recognize the order of the input sequence and consists of several transformer blocks. Self-attention calculates the relationships between words in the input sequence to determine how each word is related to other words. Additionally, the feedforward network is applied independently to each location after the self-attention process. There is research on power indices based on average representation [3]. This study obtains a pair of new power indices by assigning equal power to all members of an equivalent class.

The following Table 1 illustrates the core components of the ST5-XXL model. In Table 1, the self-attention mechanism is represented by the function Attention (Q, K, V), where Q denotes the Query vector, which transforms each word in the input sequence into a query vector. Similarly, K represents the Key vector, converting each word in the same input sequence into a key vector. V stands for the Value vector, which transforms each word in the same input sequence into a value vector. The term dk refers to the dimension of the key vector, and the use of the square root is crucial for addressing the gradient vanishing problem that can occur during training. This issue becomes more pronounced as the input sequence lengthens, leading to diminished gradient values and hindered learning. By incorporating the square root, the gradient values are maintained at a manageable level, stabilizing the learning process. Another reason for applying the square root is to stabilize the self-attention scores. When using the Softmax function, the input vector values can vary significantly, with large values potentially dominating the attention mechanism and absorbing all the attention, thereby disrupting the learning of other vectors. The square root normalizes the range of input vector values, ensuring that all vectors receive comparable attention scores and contributing to a more stable learning process.

The research utilizes multi-head attention instead of single attention. Each head independently performs self-attention, with Wo representing the learnable weight matrices WiQ, WiK, and WiV. Multi-head attention($\mathrm{Attention}(Q, K, V)$) employs multiple self-attention heads in parallel to extract diverse information from the input sentence in Equation (1) ~ (4).

Self-attention is given by

$$softmax(\frac{QK^T}{\sqrt{d_k}})\ \mathrm{V} \tag{1}$$

Each head uses different weight matrices to analyze the sentence, thus capturing the relationships between words within the sentence more comprehensively. This is the process of weighted summing the value vectors based on their relevance. This converts each word into a vector representation that reflects its contextual relationship with other words.

In Multi-head attention, $\mathrm{MultiHead}(Q, K, V)$ is given by

$$\mathrm{Attention}\left(\mathrm{Q} * W_i^Q, K * W_i^K, \mathrm{V} * W_i^V\right) \tag{2}$$

Equation (2) represents the process of converting the query, key, and value matrices of each head through a weight matrix in multi-head attention and then applying the attention mechanism. The feedforward network is represented as FFN(x), where W_1 and W_2 are weight matrices, and b_1 and b_2 are bias vectors. This network processes each position independently after self-attention, contributing to the overall transformation of the input sequence.

Here, pos denotes the position of the word, i represents the embedding dimension, and d_{model} is the total dimension of the embeddings. Positional encoding ensures that the model can capture the sequence information, which is crucial for understanding the context and meaning within the text. By leveraging these advanced techniques, the ST5-XXL model can efficiently identify and analyze potential security vulnerabilities in source code, demonstrating its exceptional performance in text classification tasks. Transformer blocks can be divided into two main components: the multi-head self-attention layer and the feedforward neural network layer.

2.1 Multi-Head Self-Attention Layer:

Represented as LayerNorm(x + MultiHead(Q, K, V)) when there are multiple heads. Here, Q, K, and V are all set to x, indicating that the query, key, and value vectors are the same as the input x.

The Layer Normalization (LayerNorm) function normalizes the input data of each layer, enhancing the stability of the learning process and ensuring faster convergence. Equation (3) is the formula used in the multihead attention of the Transformer model.

$$Concat\left(head_1, head_2, \ldots, head_h\right) * W^o \tag{3}$$

2.2 Feedforward Neural Network Layer:

Represented as LayerNorm(x + FFN(x)), the feedforward neural network (FFN(x)) is composed of two fully connected layers applied independently at each position, using a non-linear activation function to learn complex patterns.

2.2.1 Function of Layer Normalization and Feedforward Neural Network

Layer Normalization: This function normalizes the input data of each layer, increasing the stability of the learning process and enabling faster convergence. Equation (4) layer normalizes the result of residual concatenation. Layer normalization improves learning stability by normalizing the output values of each layer of the neural network.

$$\mathrm{LayerNorm}(\mathrm{x} + \mathrm{MultiHead}(\mathrm{Q}, \mathrm{K}, \mathrm{V})) \tag{4}$$

Equation (5) layer normalizes the result of residual concatenation.

$$\mathrm{LayerNorm}(\mathrm{x} + \mathrm{FFN}(\mathrm{x})) \tag{5}$$

In the Transformer model, the above two equations maintain input information through residual concatenation and layer normalization while additionally utilizing information obtained through attention and feedforward networks.

Feedforward Neural Network (FFN(x)): It consists of two fully connected layers applied independently to each position. By using a non-linear activation function, the network can learn complex patterns. Equation (6) represents the Transformer's Feed-Forward Network (FFN).

$$\mathrm{FFN}(\mathrm{x}) = ReLU(xW_1 + b_1) * w_2 + b_2 \tag{6}$$

Equation (7) represents the positional encoding value of an even index position.

$$\text{Positional Encoding } (\mathrm{PE}(\mathrm{pos}, 2\mathrm{i})) = \sin(\frac{pos}{1000^{2i/d_{model}}}) \tag{7}$$

Equation (8) represents the positional encoding value of odd index positions.

$$\text{Positional Encoding } (\mathrm{PE}(\mathrm{pos}, 2\mathrm{i}+1)) = \sin(\frac{pos}{1000^{\frac{2i}{d_{model}}}}) \tag{8}$$

It is used to encode the positional information of each word in the input sequence in the transformer model.

2.2.2 Final Output of the ST5-XXL Model

The final output of the ST5-XXL model is computed after combining the outputs of the transformer blocks and applying the softmax function to calculate the probability distribution of the next word. Thus, the final output Output is given by softmax(Wo * h), where Wo is the output weight matrix and h is the output from the last transformer block.

$$\mathrm{softmax}(w_o * \mathrm{h}) \tag{9}$$

A. Vaswani et al. proposed a transducer model based only on attention mechanisms [4]. Unlike RNN or CNN, the transformer model provides high performance without depending on the structure of circulation or convolution due to its structural characteristics. This structure operates based on an attention mechanism that allows parallel processing.

Other studies include research on Deep Residual Learning for image recognition [5]. Researchers are explicitly designing layers to learn residual functions by referencing the layer input. The dataset used by the researchers is ImageNet, a residual network with a depth of 152 layers. When using a residual block, the input value is passed to the next layer while adding learned values through several intermediate layers. This structure can solve the gradient loss problem that becomes more evident as the network depth increases, making it easier to train in an environment with sufficient network depth.

The XLNet model is a model developed to overcome the limitations of the BERT model [6]. This model combines the auto-encoding (AE) approach used in BERT for pre-train and the auto-regressive (AR) model. In the case of the AR model, it is not effective in learning bidirectional situational information by predicting the situation in the forward or backward direction. On the other hand, in the case of AE models, by masking the input and reconstructing the original data, dependencies between masked positions can be ignored, causing discrepancies between pre-train and fine tuning.

The XLNet model provides superior performance compared to the BERT model on a variety of data sets [7-9]. The Stanford Question Answering Dataset (SQuAD) is a dataset that aims to find answers to natural language questions. ReAding Comprehension from Examinations (RACE) is a dataset based on English reading test questions. General Language Understanding Evaluation (GLUE) is a dataset for natural language understanding tasks. The Yelp review dataset is used for sentiment analysis based on Yelp reviews. Amazon Review Dataset is used for sentiment analysis based on Amazon reviews.

The XLNet model, which utilizes the strengths of both the auto-regressive model and the auto-encoding model, shows excellent performance in natural language understanding and question answering tasks on the various data sets mentioned above. In this paper, the XLNet model is applied to vulnerability classification.

In Equation (10) ~ (11), It shown the core components of the XLNet model. XLNet combines both forward and backward language modeling, represented as Autoregressive Language Modeling in Table 1. Here, p(x) denotes the probability of the entire sequence, and each token x_t has a conditional probability given the preceding tokens $x_{<t}$. XLNet, like the BERT model, employs a mechanism where tokens in the input sequence are hidden, and the model predicts other tokens. The autoencoding (AE) representation in Table 1 indicates that x_{-t} represents all input tokens except the one at position t. h_t is the representation of the token at the t-th position. π represents a permutation of a given sequence, and $x_{\pi(t)}$ denotes the t-th token in the permuted sequence. This allows predictions for all token positions through various permutations.

In the context of autoregressive (AR) language modeling, the interpretation of the equation log $p_\theta(x)$ is to maximize the log-likelihood of a given sentence x. The term $\sum_{t=1}^{T}\log p_\theta(x_t|x_{<t})$ sums the log probabilities that the current word x_t is predicted by the preceding words $x_{<t}$ at each time step t of the sequence. $h_\theta(x_{1:t-1})$ represents the context representation for the sequence $x_{1:t-1}$, generated by a neural network such as an RNN or Transformer. $e(x_t)$ is the embedding of the word x_t. $\exp(h_\theta(x_{1:t-1})^T e(x_t))$ indicates the probability that the current word x_t is chosen given the context $x_{1:t-1}$. The term $\sum_{x'}\exp(h_\theta(x_{1:t-1})^T e(x'))$ acts as a normalization factor, representing the sum of probabilities for all possible words x' in the context $x_{1:t-1}$.

Equation (10) represents a probability distribution frequently used in sequence models.

$$p(x) = \Pi_{t=1}^{T} p(x_t \,|x_{<t}) \tag{10}$$

The interpretation of BERT's AE-based pre-training formula *log* $p_\theta(\bar{x}|\hat{x})$ is to maximize the log-likelihood of reconstructing the original data $\bar{x}$ from the corrupted input $\hat{x}$. The mask vector m_t is 1 if the word x_i is masked, and 0 otherwise, indicating which words are masked. Equation (10) expresses the log probability of a specific element x_t of sequence data conditioned on the remaining sequence x_{-t} excluding that element.

$$\mathrm{AE(x)} = \log P(x_t|x_{-t}) \tag{11}$$

In BERT, 15% of words in the input sentence are randomly replaced with a mask token. The model learns to predict the correct words at the masked positions by taking the masked sentence as input. $\bar{x}$ is the original sentence, and $\hat{x}$ is the masked input sentence. m_t indicates that the t-th word is masked.

$H_\theta(\hat{x})_t$ is the hidden vector for the t-th position in the masked input sentence, while $e(x_t)$ is the embedding of the t-th word. The hidden vector $H_\theta(\hat{x})$ represents the hidden vectors for each position t of the corrupted input $\hat{x}$, generated by the transformer model. Therefore, the equation illustrates the process of maximizing the log-likelihood of the original words in the masked input sentence.

The term $\sum x'\exp(H_\theta(\hat{x})_t^T e(x_t))$ represents the exponential function of the dot product between the hidden vector $H_\theta(\hat{x})_t$ of the corrupted input $\hat{x}$ and the embedding $e(x_t)$ of the current word x_t. This calculation determines the probability that the current word x_t is selected given the context $\hat{x}$. The term $\sum x'\exp(H_\theta(\hat{x})_t^T e(x'))$ signifies the sum of the exponential functions of the dot product between the hidden vector $H_\theta(\hat{x})_t$ of the corrupted input $\hat{x}$ and the embedding $e(x')$ of each possible word x'. This acts as a normalization factor for the probability. Here, $E_{Z\sim ZT}$ denotes the expected value over all possible index permutations Z_T. The term $\sum_{t=1}^T \log p_\theta(x_{zt}|x_{z<t})$ sums the log probabilities that the current word x_{zt} at each time step t is predicted by the preceding words $x_{z<t}$ according to the permutation z. This approach allows the model to learn from all possible bidirectional contexts. In Permutation language modeling in the XLNet model, $E_{Z\sim ZT}$ denotes the expected value over all possible index permutations Z_T. The term $\sum_{t=1}^T \log p_\theta(x_{zt}|x_{z<t})$ sums the log probabilities that the current word x_{zt} at each time step t is predicted by the preceding words $x_{z<t}$ according to the permutation z. This approach enables the model to learn from all possible bidirectional contexts. Equation (11) is a form of loss function used in Softmax Regression, which evaluates how well the target xt is predicted for a given input x^. By optimizing θ, the model will have better predictive performance for the given data.

$$\max_\theta \log p_\theta(\bar{x}|\hat{x}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t|\hat{x}) = \sum_{t=1}^{T} m_t \log \frac{\exp(H_0(\hat{x})^T e(x_t))}{\sum_{x'} \exp(H_0(\hat{x})^T e(x'))} \tag{11}$$

Z. Yang and colleagues argue that, unlike BERT, the XLNet model uses permutation language modeling to maximize the log-likelihood of all possible sequence orders [10]. Through this technique, XLNet can learn bidirectional context at each position without corrupting the data. While BERT is trained to restore the original data by masking and predicting certain tokens, XLNet learns contextual information in all directions naturally without data corruption. Moreover, XLNet employs an autoregressive modeling approach, which helps resolve the discrepancy between pre-training and fine-tuning phases.

3. Experiments and Results

This chapter describes about feature of experiments used data sets, workflow of XLNet and ST5-XXL model.

3.1 Features of data sets used experiments

The dataset(https://www.kaggle.com/datasets/ramoliyafenil/text-based-cyber-threat-detection) used in the experiments is described in Table 3. In this dataset, each feature serves a specific purpose in identifying and managing vulnerabilities. Below is a detailed explanation of each feature and its role:

Index: This is a unique identifier for each data entry, used to uniquely identify and manage the data for tracking and maintenance purposes.

Table 3. Features of data sets for train

index	text	entities	relations	comments	id	label	start_offset	end_offset…
1			[]	[]	45800	malware	288	300
2		[]						
3								
..	The first known..							

ID: A unique identifier for each instance within the dataset, ensuring consistency and integrity of the data within the database by uniquely identifying each entry.

Text: This includes text content transmitted over networks, such as emails, messages, and network traffic payloads. As a key data type to be analyzed, it contains natural language text with potential vulnerabilities. It is used as input for machine learning models to determine the classification of vulnerabilities.

Entities: This field contains a JSON list of entities including sender_id, label, start_offset, end_offset, receiver_ids, relations, diagnosis, and solution.

Sender_id: The ID of the entity initiating the communication or transmission.

Label: The type of identified cyber threat or attack pattern, such as malware, attack patterns, IDs, positives, software attacks, or threat actors. It represents the classification label for the text data, indicating the result of the vulnerability classification assigned to each text entry.

Start Offset: The starting character position of the identified entity or threat within the text field, helping to pinpoint the exact location of critical information and enhance the accuracy of text analysis.

End Offset: The ending character position of the identified entity or threat within the text field.

Receiver_ids: The IDs of entities receiving the communication or the intended recipients.

Relations: A list of tuples representing relationships between entities, including pairs of entity IDs indicating the source and target of the relationship. This helps understand interactions between entities, which is crucial for contextualizing vulnerabilities, such as how a vulnerability relates to specific system components.

Comments: Additional annotations or explanations providing extra information about the data entries to assist with analysis and interpretation.

Diagnosis: Descriptions or diagnoses of identified cyber threats, offering insights into the nature and potential impact of the threats.

Solution: Mitigation strategies such as specific security controls, software updates, or network configuration implementations.

3.2 Workflow of XLNet & ST5-XXL model

Figure 1 visualizes the entire workflow for data processing and model training, systematically detailing the process from data collection to model saving.

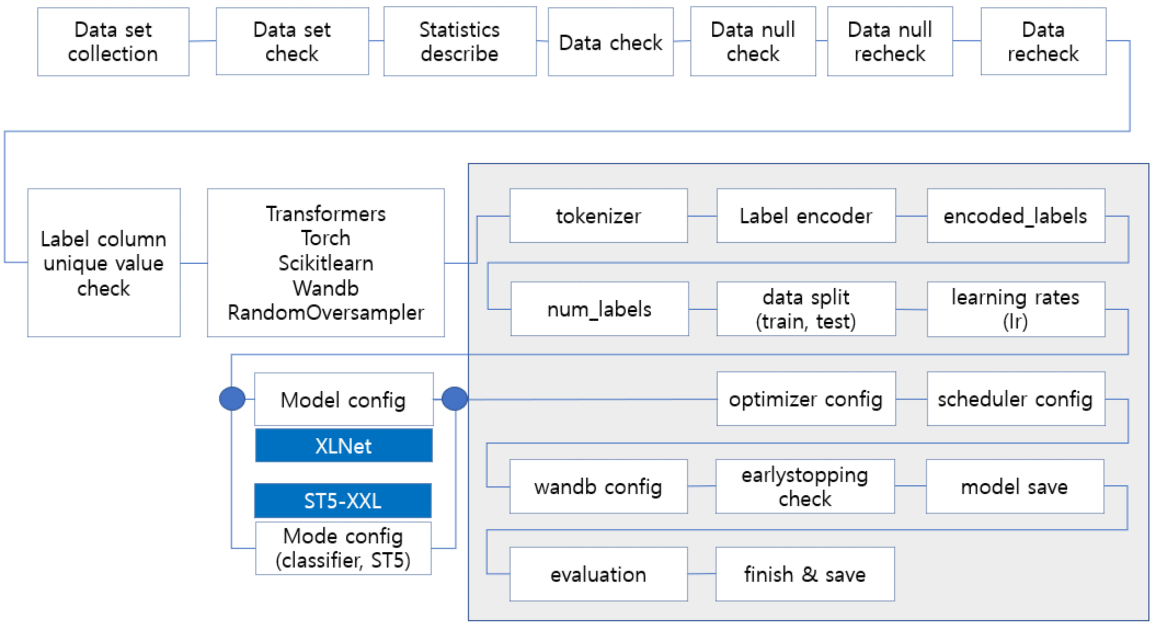


Figure 1. Work flow for learning of XLNet or ST5-XXL model

3.2.1 Data Processing Stage

Data Set Collection: This is the initial stage where datasets are collected.
Data Set Check: In this stage, the collected datasets are reviewed.
Statistics Describe: This involves describing the statistical information of the dataset.
Data Check: Here, the quality of the dataset is re-verified.
Data Null Check: This stage checks for any missing values in the dataset.
Data Null Recheck: A re-verification to ensure no missing values are overlooked.
Data Recheck: The final check of the entire dataset.

3.2.2 Data Preprocessing and Model Configuration Stage

Label Column Unique Value Check: This step verifies the uniqueness of values in the label column.

Tool and Library Utilization: Using various libraries and tools like Transformers, Torch, Scikit-learn, Wandb, and RandomOversampler to prepare for data preprocessing and model training.

3.2.3 Detailed Configuration Stage

Model Configuration:

XLNet Configuration: Setting up the XLNet model.
ST5-XXL Configuration: Setting up the ST5-XXL model.
Mode Configuration (classifier, ST5): Configuring the mode for either the classifier or the ST5 model.
Tokenizer: Tokenizing the text data using a tokenizer.
Label Encoder: Converting labels into numerical form using a label encoder.
Encoded Labels: The resulting encoded labels.
Num Labels: Setting the number of labels.
Data Split (train, test): Dividing the data into training and testing sets.
Learning Rates (lr): Setting the learning rates for the models.

3.2.4 Training Configuration Stage

Optimizer Configuration: Setting up the optimizer.
Scheduler Configuration: Configuring the learning scheduler.
Weights & Biases (Wandb) Configuration: Setting up Weights & Biases (Wandb).
Early Stopping Check: Checking the conditions for early stopping.
Model Save: Saving the trained model.

3.2.5 Evaluation and Saving Stage

Evaluation: Evaluating the trained model.
Finish & Save: Saving the final results and concluding the process.

This workflow encompasses all stages from data collection to model training, evaluation, and saving, illustrating a systematic and consistent approach to conducting a data science project.

3.3 Experimental Results

Figure 2 compares the performance of two models, XLNet and ST5-XXL, at various learning rates. Each bar represents the model's performance at a specific learning rate, with the performance percentage displayed in parentheses. The learning rates are set at 1e-06, 1e-05, and 1e-04.

ST5-XXL_model_1e-06: Achieves a performance of 91.675% at a learning rate of 1e-06.
ST5-XXL_model_1e-05: Achieves a performance of 93.581% at a learning rate of 1e-05.
ST5-XXL_model_1e-04: Achieves a performance of 93.681% at a learning rate of 1e-04.
XLNet_model_lr_1e-06: Achieves a performance of 82.321% at a learning rate of 1e-06.
XLNet_model_lr_1e-05: Achieves a performance of 95.387% at a learning rate of 1e-05.
XLNet_model_lr_1e-04: Achieves a performance of 94.742% at a learning rate of 1e-04.

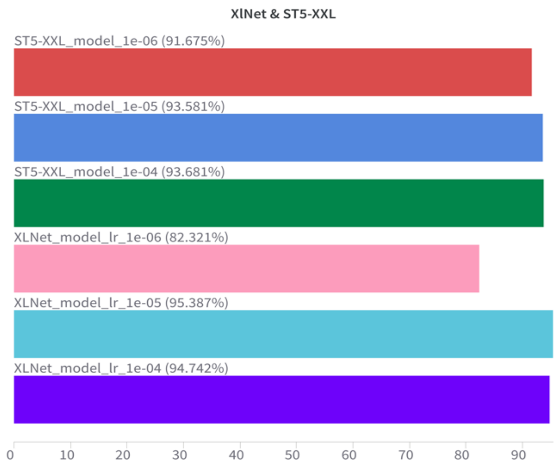


Figure 2. Accuracy of XLNet and ST5-XXL model

Considering the impact of learning rates, it is evident that learning rates significantly affect the performance of both models. Particularly for XLNet, the highest performance is observed at a learning rate of 1e-05. When comparing the models, XLNet generally outperforms ST5-XXL at the same learning rates, demonstrating superior performance across the board.

Figure 3 and its legend depict the change in training loss for XLNet and ST5-XXL models over training steps. The graph visually demonstrates how each model learns at different learning rates.

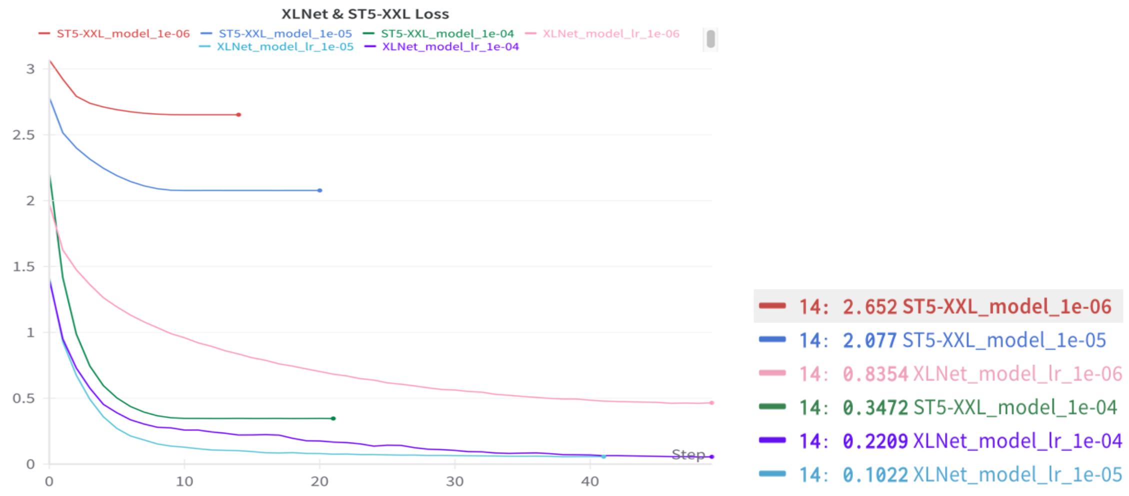


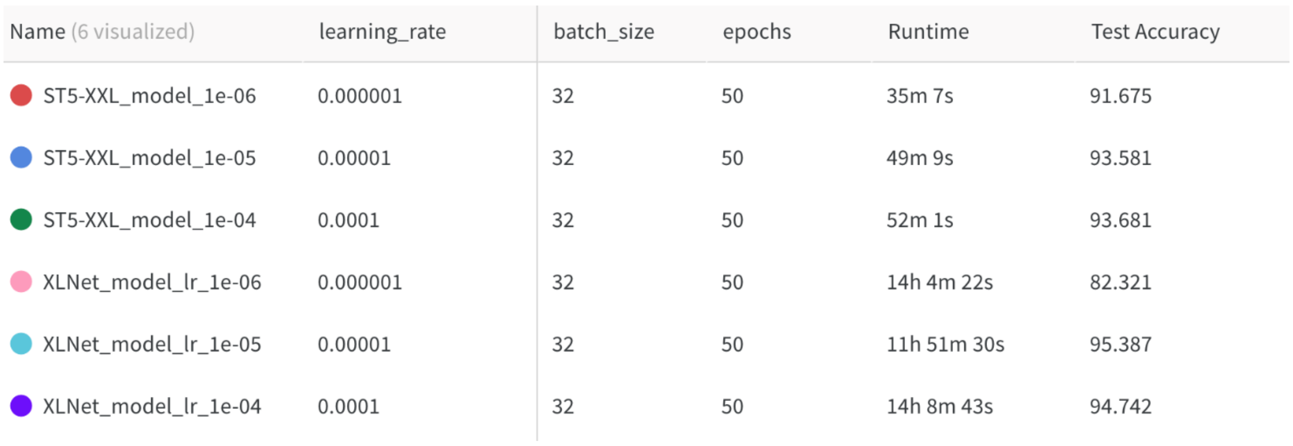
Figure 3. Loss rate of XLNet and ST5-XXL model

This detailed visualization helps in understanding the convergence behavior of each model and how effectively they minimize loss during training at various learning rates. The analysis focuses on the impact of learning rates on training efficiency and model performance. By comparing the learning curves, one can discern the optimal learning rate for each model, providing insights into the training dynamics and stability of XLNet and ST5-XXL under different configurations.

Table 4 shows about performance comparison in XLNet and ST5-XXL model. The **ST5-XXL_model_1e-**

06 (red) starts with an initial loss value of around 3. Although the loss decreases as training progresses, it maintains a higher loss value compared to other models. The **ST5-XXL_model_1e-05** (blue) begins with an initial loss of approximately 2 and decreases rapidly, but it learns slower than the other models. The **ST5-XXL_model_1e-04** (green) also starts with an initial loss of about 2 and decreases very quickly, showing the lowest loss values among the ST5-XXL models.

Table 4. Performance comparison in according to model



Name (6 visualized)	learning_rate	batch_size	epochs	Runtime	Test Accuracy
● ST5-XXL_model_1e-06	0.000001	32	50	35m 7s	91.675
● ST5-XXL_model_1e-05	0.00001	32	50	49m 9s	93.581
● ST5-XXL_model_1e-04	0.0001	32	50	52m 1s	93.681
● XLNet_model_lr_1e-06	0.000001	32	50	14h 4m 22s	82.321
● XLNet_model_lr_1e-05	0.00001	32	50	11h 51m 30s	95.387
● XLNet_model_lr_1e-04	0.0001	32	50	14h 8m 43s	94.742

For the XLNet models, the **XLNet_model_lr_1e-06** (pink) starts with an initial loss of around 2 and gradually decreases over time, but it maintains higher loss values compared to the other models. The **XLNet_model_lr_1e-05** (purple) starts with an initial loss of around 2 and decreases very quickly, showing the second lowest loss values next to the 1e-04 learning rate model. The **XLNet_model_lr_1e-04** (cyan) begins with an initial loss of around 2 and decreases the fastest, maintaining the lowest loss values overall.

The impact of learning rates on both models is significant, as the loss values change considerably with different learning rates. Specifically, the XLNet model performs exceptionally well at learning rates of 1e-05 and 1e-04. In comparison, the XLNet model generally demonstrates lower loss values than the ST5-XXL model at the same learning rates.

Finding an appropriate learning rate is crucial, as too low or too high a learning rate can result in slow or unstable training. The optimal learning rate ensures efficient and stable learning, minimizing the loss effectively.

4. Conclusion

We analyzed the data processing and model training procedures in detail to determine the level of discomfort using the XLNet and ST5-XXL models. The dataset used in the experiment contains various features such as unique extracts, text data, entity information, relationships, descriptions, and labels to distinguish them. In our experiments, we confirmed that the XLNet model showed excellent performance at learning rates of 1e-05 and 1e-04, consistently showing lower values than the ST5-XXL model. XLNet models learn efficiently and have greater potential. As a result of analyzing the problem of training loss, it is an object whose value changes significantly depending on the research results of XLNet and ST5-XXL models. In particular, the XLNet model shows that the decline is further accelerated. Comparatively, the ST5-XXL model has relatively superior speed by maintaining a smaller value at a certain ratio. We argue that setting an appropriate learning rate is important to distinguish between underfitting and underfitting of model training. The XLNet model can be an effective tool for specialized tasks as it can show high performance at various learning rates, especially 1e-05 and 1e-04. We verified the performance of ST5-XXL and XLNet models in experiments, and predict that future research should include various experiments using various data sets and additional models.

Reference

[1] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, et al., PaLM: Scaling Language Modeling with Pathways," pp. 1-87, Apr. 2022. *https://arxiv.org/abs/2204.02311*. DOI: https://doi.org/10.48550/arXiv.2204.02311

[2] J. Ni, G. H. Abrego, N. Constant, J. Ma, K. Hall, D. Cer, Y. Yang, "Sentence-T5: Scalable Sentence Encoder s from Pre-trained Text-to-Text Models," in *Proc. ACL*, pp. 1864-1874, May 22-27, 2022. *https://aclanthology.org/2022.findings-acl.146*. DOI: http://doi.org/10.18653/v1/2022.findings-acl.146.

[3] S. Kaniovski, S. Kurz, "Representaiton-Compatible Power Indices," Springer, Vol. 264, No. 1, pp. 235-265, May 2018. *https://arxiv.org/abs/1506.05963*. DOI: http://doi.org/10.1007/s10479-017-2672-3.

[4] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition," pp. 1-12, Dec. 2015. *https://arxiv.org/abs/1512.03385*. DOI: https://doi.org/10.48550/arXiv.1512.03385.

[5] F. Ramoliya, R. Kakkar, R. Gupta, S. Tanwar, S. Agrawai, "SEAM: Deep Learning-based Secure Message Exchange Framework for Autonomous EVs," in *Proc. IEEE Globecom Workshops*, pp. 80-85, March 21, 2023. DOI: https://doi.org/10.1109/GCWkshps58843.2023.10465168.

[6] J. Ni, G. H. Abrego, N. Constant, J. Ma, K. Hall, aD. Cer, Y. Yang, "Sentence-T5: Scalable Encoders from Pre-trained Text-to-Text Models," in *Proc. ACL*, pp.1864-1874, May 22-27, 2022.

[7] A. Vaswani, N. Shazeer , N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, "Attention is All You Need," in *Proc. 31st conference on NIPS*, pp. 1-15, Aug. 2023. *https://arxiv.org/abs/1706.03762*. DOI: http://doi.org/10.48550/arXiv.1706.03762.

[8] https://www.kaggle.com/datasets/ramoliyafenil/text-based-cyber-threat-detection

[9] S.-S. Kim, "Deep Learning-Based Inverse Design for Engineering Systems: A Study on Supervised and Unsupervised Learning Models," *Journal of Internet, Broadcasting and Communication*, Vol. 16, No. 2, pp. 127-135, May 2024. DOI: http://dx.doi.org/10.7236/IJIBC.2024.16.2.127.

[10] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salahutdinov, Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," in *Proc. 33rd conference on NeurIPS*, pp. 1-18, Jan. 2020. *https://arxiv.org/abs/1906.08237v2*. DOI: https://doi.org/10.48550/arXiv.1906.08237.