NLP ASSIGNMENT 2
REPORT

NAME: SRUJAN ALUMULA
UID: U42726372

- **Introduction: Objectives and overview of the work.**

Analysing and contrasting the effectiveness of character-level and word-level tokenization techniques on a text classification task using the IMDB dataset is the main goal of this work.
Experimenting with different hyperparameters, such as activation functions, learning rates, batch sizes, and optimizers, is also intended to maximize the performance of the model.

Tokenization plays a crucial role in the text preprocessing significantly impacting the model performance. This study compares the two tokenization approaches to understand their influence on the model's ability to classify the text data.

Hyperparameter optimization is used systematically to determine the best combination of parameters for achieving the better high accuracy.

- **Methodology: Detailed explanation of tokenization changes and hyper-parameter optimization strategy**.

  Tokenization Changes :Character Level vs Word Level
  The main focus was on two distinct types that are character level tokenization and word level tokenization. The main objective was to compare their impact on the model performance, training efficiency and the final evaluation metrics.
  Character Level Tokenization
  Character level tokenization involves breaking down text into the individual characters, rather than the whole words. This approach is especially useful for handling rare words, misspellings, or language with complex word structures. In this project the following steps were followed for character level tokenization.

  Text Preprocessing : The text data first preprocessed to convert all the characters to lowercase ensuring the consistency across inputs.
  Tokenizer Setup: A character level tokenizer was built using the tensorflow tokenizer api with the charlevel = true parameter. This tokenizer was then fitted on the training data to learn the vocabulary of characters.
  Bag of Characters – After Tokeinzation the text was transformed into the binary bag of characters representation using the matrix. This matrix represented the presence or absence of each character in the vocabulary.

  Hyperparameters Optimized:

Activation Function : activation function is the how the weighted sum of inputs is transformed into an output. Serval activation functions were tested
Leaky relu – A variant of relu that allows small nonzero gradients when the input is negative potentially preventing dead neurons.

Relu : A widely used activation function that outputs zero for negative inputs and the same value for the positive inputs.

Tanh : A squashed activation function with outputs between -1 and 1 often used the data distribution is centered around 0.

Optimizer : The optimizer is responsible for the adjusting the weights during training . The following optimizers were I tested.

Adam: An adaptive optimizer that computes adaptive learning rated for each parameter.

SGD: Stochastic Gradient Descent, a simple optimizer often used with momentum for faster convergence.

RMSprop: An optimizer that adjusts the learning rate for each parameter based on the recent gradient information.

- And Also the remaining all I tested with these
  Learning Rate: [0.001, 0.0005, 0.0001]
- **Hidden Layers:** [1, 2, 3]
- **Hidden Sizes:** [128, 256, 512]
- **Batch Sizes:** [32, 64, 128]
- **Optimizers:** [Adam, SGD, RMSProp]
- **Activation Functions:** [ReLU, Tanh, LeakyReLU]

SEED	Activation Function	Optimizer	Batch Size	Learning Rate	Testing Accuracy	Test Loss
655	Relu	RMSProp	512	0.0001	60.37	66.35
655	Relu	RMSProp	128	0.0001	60.78	66.06
655	Relu	SGD	128	0.0001	53.39	69.07
655	Tanh	Adam	512	0.0001	60.44	66.12
655	LeakyRelu	Adam	512	0.0001	60.44	66.25
655	Relu	Adam	512	0.0001	60.64	66.28
655	Relu	Adam	256	0.001	60.97	65.77
655	Relu	Adam	256	0.0005	60.65	66.11

655	Relu	Adam	256	0.0001	60.75	66.06
655	Relu	Adam	256	0.001	60.66	66.14
655	Relu	Adam	256	0.001	60.64	66.08
655	Relu	Adam	256	0.0001	60.51	66.15
655	Relu	Adam	256	0.0005	60.88	65.90
655	Tanh	Adam	128	0.0001	60.44	66.12
655	Relu	Adam	128	0.001	60.64	66.14
655	Relu	SGD	512	0.0001	60.22	70.54
655	Tanh	Adam	512	0.001	60.68	65.95
655	LeakyRelu	Adam	512	0.001	61.01	65.79
655	LeakyRelu	SGD	512	0.01	53.90	68.91
655	LeakyRelu	RMSProp	512	0.01	56.90	68.41
655	LeakyRelu	Adam	256	0.01	61.04	65.94
655	LeakyRelu	Adam	128	0.01	60.64	66.11
1567	LeakyRelu	Adam	256	0.01	58.74	60.81
45	LeakyRelu	Adam	256	0.01	61.14	65.83

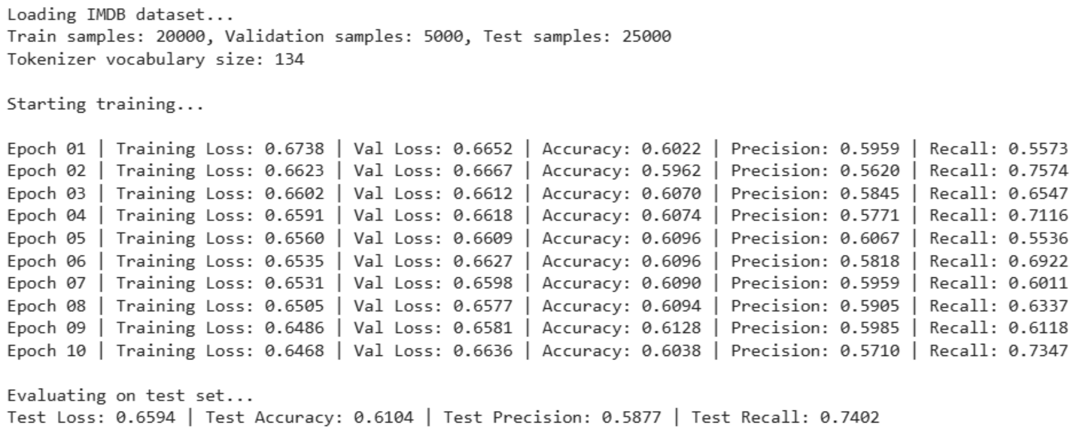
655 (RANDOM) model	LeakyRelu	Adam	256	0.01	58.52	67.29
45	LeakyRelu	Adam	256	0.01	59.10	67.07

Observation :
Among the activation function tested the LeakyRelu is performed consistently well achieving the highest accuracy of 61.16% with a test loss of 65.83 at 256 batch size and learning rate of 0.01.

Optimizer : Adam appeared to be the most reliable optimizer across various configurations producing higher accuracy results compared to RMSProp or SGD.
Batch Size : Larger batches generally resulted in lower accuracy and higher loss compared to medium batches sizes with the leakyrelu and adam optimizer.

Learning Rate: Learning rate around 0.01 consistenly showed better results than smaller rates with the higher learning rated tending to achieve slightly better performance.

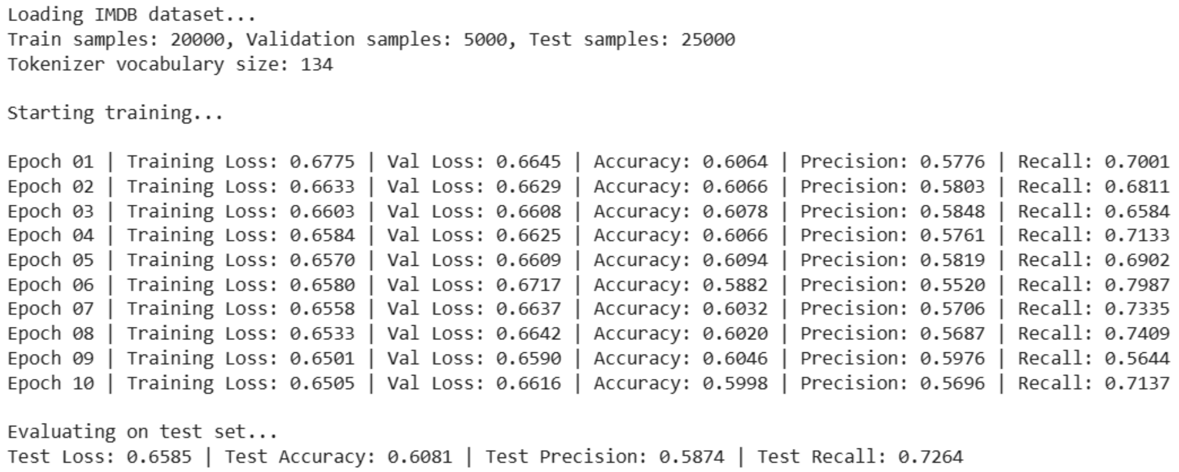
This Is the Best I verified from the above all
with seed 655 , activation function Leaky Relu , Optimizer Adam , batch Size =256 Learning Rate 0.01



```
Loading IMDB dataset...
Train samples: 20000, Validation samples: 5000, Test samples: 25000
Tokenizer vocabulary size: 134

Starting training...

Epoch 01 | Training Loss: 0.6738 | Val Loss: 0.6652 | Accuracy: 0.6022 | Precision: 0.5959 | Recall: 0.5573
Epoch 02 | Training Loss: 0.6623 | Val Loss: 0.6667 | Accuracy: 0.5962 | Precision: 0.5620 | Recall: 0.7574
Epoch 03 | Training Loss: 0.6602 | Val Loss: 0.6612 | Accuracy: 0.6070 | Precision: 0.5845 | Recall: 0.6547
Epoch 04 | Training Loss: 0.6591 | Val Loss: 0.6618 | Accuracy: 0.6074 | Precision: 0.5771 | Recall: 0.7116
Epoch 05 | Training Loss: 0.6560 | Val Loss: 0.6609 | Accuracy: 0.6096 | Precision: 0.6067 | Recall: 0.5536
Epoch 06 | Training Loss: 0.6535 | Val Loss: 0.6627 | Accuracy: 0.6096 | Precision: 0.5818 | Recall: 0.6922
Epoch 07 | Training Loss: 0.6531 | Val Loss: 0.6598 | Accuracy: 0.6090 | Precision: 0.5959 | Recall: 0.6011
Epoch 08 | Training Loss: 0.6505 | Val Loss: 0.6577 | Accuracy: 0.6094 | Precision: 0.5905 | Recall: 0.6337
Epoch 09 | Training Loss: 0.6486 | Val Loss: 0.6581 | Accuracy: 0.6128 | Precision: 0.5985 | Recall: 0.6118
Epoch 10 | Training Loss: 0.6468 | Val Loss: 0.6636 | Accuracy: 0.6038 | Precision: 0.5710 | Recall: 0.7347

Evaluating on test set...
Test Loss: 0.6594 | Test Accuracy: 0.6104 | Test Precision: 0.5877 | Test Recall: 0.7402
```

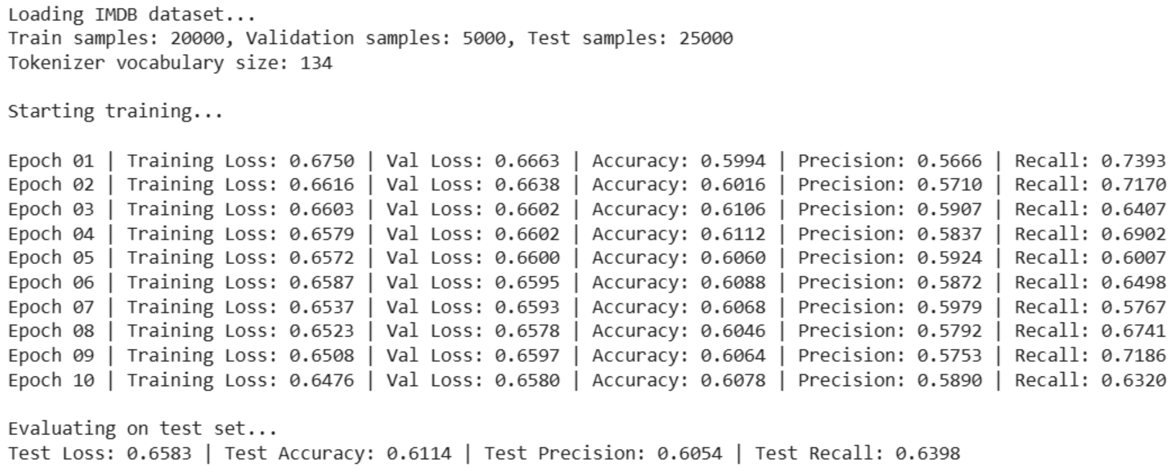
These Model is verified with Seed Numbers

Test 2 with - Seed Number 1567



```
Loading IMDB dataset...
Train samples: 20000, Validation samples: 5000, Test samples: 25000
Tokenizer vocabulary size: 134

Starting training...

Epoch 01 | Training Loss: 0.6775 | Val Loss: 0.6645 | Accuracy: 0.6064 | Precision: 0.5776 | Recall: 0.7001
Epoch 02 | Training Loss: 0.6633 | Val Loss: 0.6629 | Accuracy: 0.6066 | Precision: 0.5803 | Recall: 0.6811
Epoch 03 | Training Loss: 0.6603 | Val Loss: 0.6608 | Accuracy: 0.6078 | Precision: 0.5848 | Recall: 0.6584
Epoch 04 | Training Loss: 0.6584 | Val Loss: 0.6625 | Accuracy: 0.6066 | Precision: 0.5761 | Recall: 0.7133
Epoch 05 | Training Loss: 0.6570 | Val Loss: 0.6609 | Accuracy: 0.6094 | Precision: 0.5819 | Recall: 0.6902
Epoch 06 | Training Loss: 0.6580 | Val Loss: 0.6717 | Accuracy: 0.5882 | Precision: 0.5520 | Recall: 0.7987
Epoch 07 | Training Loss: 0.6558 | Val Loss: 0.6637 | Accuracy: 0.6032 | Precision: 0.5706 | Recall: 0.7335
Epoch 08 | Training Loss: 0.6533 | Val Loss: 0.6642 | Accuracy: 0.6020 | Precision: 0.5687 | Recall: 0.7409
Epoch 09 | Training Loss: 0.6501 | Val Loss: 0.6590 | Accuracy: 0.6046 | Precision: 0.5976 | Recall: 0.5644
Epoch 10 | Training Loss: 0.6505 | Val Loss: 0.6616 | Accuracy: 0.5998 | Precision: 0.5696 | Recall: 0.7137

Evaluating on test set...
Test Loss: 0.6585 | Test Accuracy: 0.6081 | Test Precision: 0.5874 | Test Recall: 0.7264
```

Test 3 With seed Number – 45

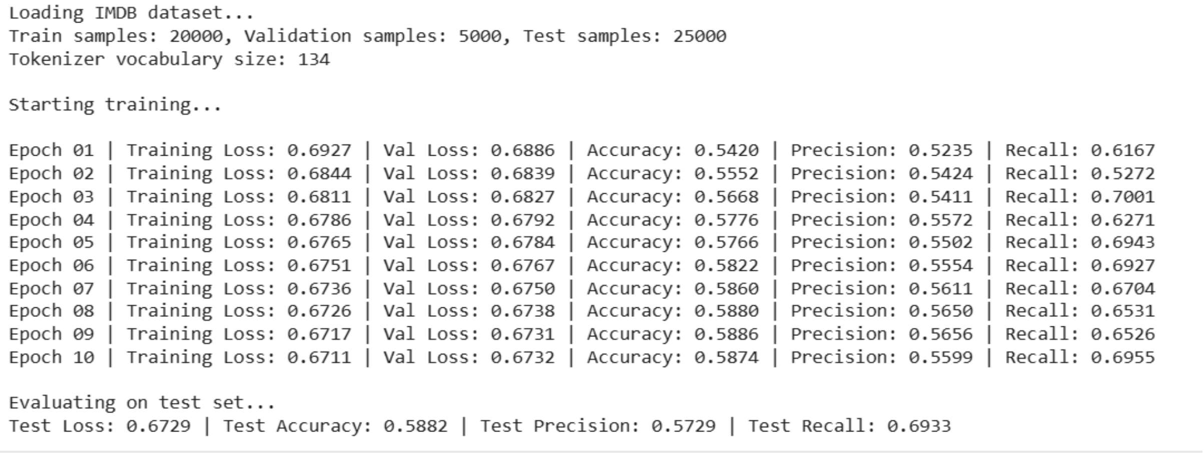


```
Loading IMDB dataset...
Train samples: 20000, Validation samples: 5000, Test samples: 25000
Tokenizer vocabulary size: 134

Starting training...

Epoch 01 | Training Loss: 0.6750 | Val Loss: 0.6663 | Accuracy: 0.5994 | Precision: 0.5666 | Recall: 0.7393
Epoch 02 | Training Loss: 0.6616 | Val Loss: 0.6638 | Accuracy: 0.6016 | Precision: 0.5710 | Recall: 0.7170
Epoch 03 | Training Loss: 0.6603 | Val Loss: 0.6602 | Accuracy: 0.6106 | Precision: 0.5907 | Recall: 0.6407
Epoch 04 | Training Loss: 0.6579 | Val Loss: 0.6602 | Accuracy: 0.6112 | Precision: 0.5837 | Recall: 0.6902
Epoch 05 | Training Loss: 0.6572 | Val Loss: 0.6600 | Accuracy: 0.6060 | Precision: 0.5924 | Recall: 0.6007
Epoch 06 | Training Loss: 0.6587 | Val Loss: 0.6595 | Accuracy: 0.6088 | Precision: 0.5872 | Recall: 0.6498
Epoch 07 | Training Loss: 0.6537 | Val Loss: 0.6593 | Accuracy: 0.6068 | Precision: 0.5979 | Recall: 0.5767
Epoch 08 | Training Loss: 0.6523 | Val Loss: 0.6578 | Accuracy: 0.6046 | Precision: 0.5792 | Recall: 0.6741
Epoch 09 | Training Loss: 0.6508 | Val Loss: 0.6597 | Accuracy: 0.6064 | Precision: 0.5753 | Recall: 0.7186
Epoch 10 | Training Loss: 0.6476 | Val Loss: 0.6580 | Accuracy: 0.6078 | Precision: 0.5890 | Recall: 0.6320

Evaluating on test set...
Test Loss: 0.6583 | Test Accuracy: 0.6114 | Test Precision: 0.6054 | Test Recall: 0.6398
```

Now These Model is compared with the Random model

Random MLP on IMDB Dataset with seed number of 655

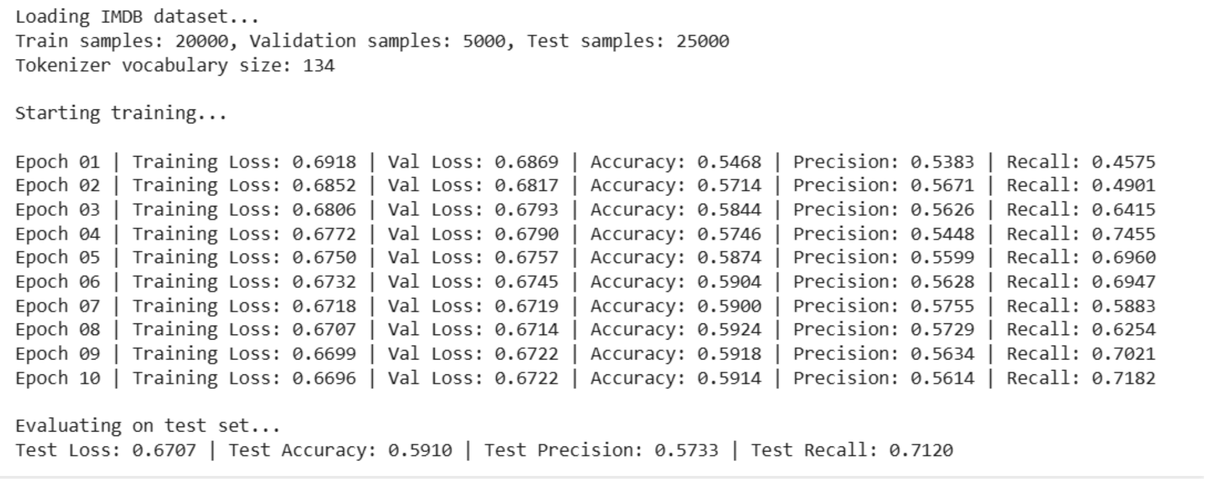
Learning Rate of 0.001 and actiation function leakyrelu with an optimizer ADAM , batach size=256



```
Loading IMDB dataset...
Train samples: 20000, Validation samples: 5000, Test samples: 25000
Tokenizer vocabulary size: 134

Starting training...

Epoch 01 | Training Loss: 0.6927 | Val Loss: 0.6886 | Accuracy: 0.5420 | Precision: 0.5235 | Recall: 0.6167
Epoch 02 | Training Loss: 0.6844 | Val Loss: 0.6839 | Accuracy: 0.5552 | Precision: 0.5424 | Recall: 0.5272
Epoch 03 | Training Loss: 0.6811 | Val Loss: 0.6827 | Accuracy: 0.5668 | Precision: 0.5411 | Recall: 0.7001
Epoch 04 | Training Loss: 0.6786 | Val Loss: 0.6792 | Accuracy: 0.5776 | Precision: 0.5572 | Recall: 0.6271
Epoch 05 | Training Loss: 0.6765 | Val Loss: 0.6784 | Accuracy: 0.5766 | Precision: 0.5502 | Recall: 0.6943
Epoch 06 | Training Loss: 0.6751 | Val Loss: 0.6767 | Accuracy: 0.5822 | Precision: 0.5554 | Recall: 0.6927
Epoch 07 | Training Loss: 0.6736 | Val Loss: 0.6750 | Accuracy: 0.5860 | Precision: 0.5611 | Recall: 0.6704
Epoch 08 | Training Loss: 0.6726 | Val Loss: 0.6738 | Accuracy: 0.5880 | Precision: 0.5650 | Recall: 0.6531
Epoch 09 | Training Loss: 0.6717 | Val Loss: 0.6731 | Accuracy: 0.5886 | Precision: 0.5656 | Recall: 0.6526
Epoch 10 | Training Loss: 0.6711 | Val Loss: 0.6732 | Accuracy: 0.5874 | Precision: 0.5599 | Recall: 0.6955

Evaluating on test set...
Test Loss: 0.6729 | Test Accuracy: 0.5882 | Test Precision: 0.5729 | Test Recall: 0.6933
```

Random MLP on IMDB Dataset with seed number of 45

Learning Rate of 0.001 and actiation function leakyrelu with an optimizer ADAM , batach size=256



```
Loading IMDB dataset...
Train samples: 20000, Validation samples: 5000, Test samples: 25000
Tokenizer vocabulary size: 134

Starting training...

Epoch 01 | Training Loss: 0.6918 | Val Loss: 0.6869 | Accuracy: 0.5468 | Precision: 0.5383 | Recall: 0.4575
Epoch 02 | Training Loss: 0.6852 | Val Loss: 0.6817 | Accuracy: 0.5714 | Precision: 0.5671 | Recall: 0.4901
Epoch 03 | Training Loss: 0.6806 | Val Loss: 0.6793 | Accuracy: 0.5844 | Precision: 0.5626 | Recall: 0.6415
Epoch 04 | Training Loss: 0.6772 | Val Loss: 0.6790 | Accuracy: 0.5746 | Precision: 0.5448 | Recall: 0.7455
Epoch 05 | Training Loss: 0.6750 | Val Loss: 0.6757 | Accuracy: 0.5874 | Precision: 0.5599 | Recall: 0.6960
Epoch 06 | Training Loss: 0.6732 | Val Loss: 0.6745 | Accuracy: 0.5904 | Precision: 0.5628 | Recall: 0.6947
Epoch 07 | Training Loss: 0.6718 | Val Loss: 0.6719 | Accuracy: 0.5900 | Precision: 0.5755 | Recall: 0.5883
Epoch 08 | Training Loss: 0.6707 | Val Loss: 0.6714 | Accuracy: 0.5924 | Precision: 0.5729 | Recall: 0.6254
Epoch 09 | Training Loss: 0.6699 | Val Loss: 0.6722 | Accuracy: 0.5918 | Precision: 0.5634 | Recall: 0.7021
Epoch 10 | Training Loss: 0.6696 | Val Loss: 0.6722 | Accuracy: 0.5914 | Precision: 0.5614 | Recall: 0.7182

Evaluating on test set...
Test Loss: 0.6707 | Test Accuracy: 0.5910 | Test Precision: 0.5733 | Test Recall: 0.7120
```

Conclusion:

Through the series of experiments I observed that certain configuration consistently yielded better performance. Specifically, LeakyRelu combined with the Adam optimizer provided the most stable and highest performace across various batch size and learning rated. Among the batch sizes with the learning rate consistently resulted in better accuracy and lower loss. The performance of the model was evaluated using the character level tokenization with the final model performance indicating that word level tokenization provided a more balanced.

The hyperparameter optimization strategy by adjusting activation functions optimizers and learning rate highlighted that Adam was the most effective optimizer, outerforming both RMSProp and SGD. In conclusion the experiments demonstrated the importance of selecting the right combination of hyperparameters as well as the choice between the tokenization in improving the model performance.