Evaluating BERT-Based Models for Fake News Classification on TruthSeeker Twitter Data: A Cybersecurity Perspective

Talia Berler

CSC616

Dr. Stephen Dennis

May 7, 2025

1. Introduction

At this point in the 21st century, many people get their news through social media and online news outlets. While convenient, receiving news primarily online and on social media makes modern societies vulnerable to the effects of mass disinformation proliferated by fake, or false, news posts. The spread of fake information is not only detrimental to societal function as a whole, but can be extremely dangerous if allowed to circulate unchecked. The effects of fake news proliferation online are a serious issue for cybersecurity, as mass dissemination of fake news enables malicious actors to manipulate vulnerable individuals online for any number of purposes. This issue motivated the following research on methods to efficiently and effectively address fake news online.

Due to the high volume of daily posts and the wide range of news source platforms online, the only feasible approach to intervention on a large, sustainable scale is automating fake news detection. In order to accomplish this task, this research evaluates various state-of-the-art BERT-based machine learning (ML) models on their ability to classify tweet text as real or fake news. While there exists literature on other approaches utilizing BERT-based models for the task of classifying news text and tweets as real or fake, this paper takes a novel approach of incorporating the recently released DeepSeek large language model (LLM) into the classification architecture. The author seeks to answer the following research questions:

1. Which of the selected BERT-based models performs the best in terms of time and classification performance with the task of detecting fake news tweets?
2. Does applying early stopping based on the metric of F1 score or accuracy affect model classification performance?

3. Does projecting BERT-based model embeddings to a higher dimension for processing by an LLM classification head improve classification performance over using the original fine-tuned base model embeddings with the default linear layer classification head?

2. Related Work

Previous research has confirmed the effectiveness of transformer models, particularly BERT-based (Bidirectional Encoder Representations from Transformers) models, for fake news detection. [1][2][3] Specific BERT-based models which have demonstrated high performance compared to other ML approaches in various state-of-the-art benchmark studies include the original BERT model in base and large formats, RoBERTa, DistilBERT, ALBERT, and BERTweet. [1][2][3] Studies have highlighted trade-offs in model performance, size, and computational efficiency amongst the aforementioned BERT-based models. BERT, RoBERTa, DistilBERT, and BERTweet were identified as consistently high performing models for the task of classifying online news text as real or fake. [1][3] Notably, the creators of the TruthSeeker dataset utilized in this study, concluded that BERTweet, pretrained on Twitter data, has shown strong performance on their large dataset of social media text. [3] The TruthSeeker creators also noted that ALBERT has been shown to underperform compared to other BERT-based models, with a tendency to overfit to the training data, and has thus been excluded from this study. [3] Meta-analyses suggest a shift toward transformer-based models, citing their superior F1 scores and robustness. [2] However, gaps remain in developing standardized evaluation criteria for ML models classifying online news content as real or fake, and in novel architectures which stack various forms of deep learning to improve classification results.

3. Dataset

3.1 TruthSeeker Dataset

The TruthSeeker dataset, developed by the Canadian Institute for Cybersecurity, includes 134,198 tweets associated with 1,058 unique news statements crawled from Politifact. [3] The tweets scraped from Twitter were posted between 2009 and 2022, with a character limit of 140 characters. [3] The topics covered by the news statements from Politifact and tweets include politics, general events, health, crime, and science, which provides a robust spread of topics to evaluate automated fake news detection. [3] The tweet observations were matched with news statements using keywords manually extracted from the Politifact statements by the authors of the TruthSeeker dataset. Using the Amazon Mechanical Turk Service, "3_label_majority_answer" and "5_label_majority_answer" features were generated via majority vote from Amazon Mechanical Turk "Master" workers describing the extent to which each tweet agreed or disagreed with the association news statement. [3] Data was cleaned to exclude unclear or contradictory labels prior to publication of the official TruthSeeker dataset, with about 150,000 observations. [3] The official published TruthSeeker dataset for advanced ML training contains 8 features: "author", "statement", "target", "BinaryNumTarget", "manual_keywords", "tweet", "5_label_majority_answer", "3_label_majority_answer" (Fig 1).

author	statement	target	BinaryNumTarget	manual_keywords	tweet	5_label_majority_answer	3_label_majority_answer
D.L. Davis	End of eviction moratorium means millions of A...	True	1.0	Americans, eviction moratorium	@POTUS Biden Blunders - 6 Month Update\n\nInfl...	Mostly Agree	Agree
D.L. Davis	End of eviction moratorium means millions of A...	True	1.0	Americans, eviction moratorium	@S0SickRick @Stairmaster_ @6d6f636869 Not as m...	NO MAJORITY	Agree

Figure 1. TruthSeeker dataset before feature selection and engineering

3.2 Preprocessing

To prepare the TruthSeeker dataset for use in this study, the process of developing a two-label truthfulness value described by the authors of the TruthSeeker dataset was replicated. Thus, for each observation in the dataset, a new target label was generated based on the truth value of the actual news statement combined with the extent the tweet agreed with the given news statement in the "3_label_majority_answer" feature, resulting in the new truthfulness feature called "label" (Fig 2). Once the new "label" feature was generated, all other features were dropped with the exception of the tweet feature, renamed as "text" (Fig 2).

TABLE V
TWO-LABEL CONVERSION TRUTH TABLE

Statement (T/F)	Majority Answer	Truthfulness
T	Agree	True
T	Disagree	False
F	Agree	False
F	Disagree	True

	text	label
0	@POTUS Biden Blunders - 6 Month Update\n\nInfl...	1
1	@S0SickRick @Stairmaster_ @6d6f636869 Not as m...	1
2	THE SUPREME COURT is siding with super rich pr...	1
3	@POTUS Biden Blunders\n\nBroken campaign promi...	1
4	@OhComfy I agree. The confluence of events rig...	1

Figure 2. The truthfulness assignment schema for each observation and the processed TruthSeeker dataset for model training.

After feature selection and engineering, the data was split into 70% training and 30% testing for model training and evaluation. The training set was further split with 10% reserved for validation of the training metrics. The splits were performed with a fixed random_state to ensure reproducibility. This resulted in 63% training set for model learning, 7% validation set for early stopping, and 30% test set for final model evaluation. The pandas dataframes for training, validation, and testing were converted to Hugging Face Dataset format to be compatible with Hugging Face's Trainer API. The "text" feature was then tokenized using Hugging Face's AutoTokenizer. For BERTweet specifically, the tweets were normalized before tokenization, as BERTweet is trained on normalization tweet text. [7] This normalization process involved

changing all usernames to “@USER” and converting all URLs to “HTTPURL”. Finally, the tokenized inputs were then padded and truncated to a fixed maximum length of 128 for BERTweet and 300 for BERT, RoBERTa, and DistilBERT, respectively.

3.3 Suitability for News Classification Task

The creators of the original TruthSeeker dataset did not clearly explain the reason for the reduction from 150,000 observations to slightly below 135,000 observations. Additionally, much detail was provided in regards to the Amazon Turker approach, but the explanation of assigning truthfulness to the tweets and the manner of correlating tweets to news statements could use more clarification. For these reasons, I believe that while the dataset is large in volume and thus, a good choice for training automated fake news detection models, the creation of the dataset is also lacking in some key areas of its explanation, which could prove important to the uses of the dataset for future model training and implementation. Regardless, this study was performed using TruthSeeker for its large volume of data, with a more detailed description of the methodology below.

4. Methodology

4.1 Model Architectures

Four BERT-based transformer base models were accessed via Hugging Face Hub and fine-tuned using Hugging Face’s Trainer API with the following model descriptions:

- BERT-base (google-bert/bert-base-uncased): The base BERT model is pretrained on BookCorpus, a dataset of about 11,000 books and English Wikipedia. This model has

110M parameters, 12 layers, 768 hidden layers, 12 attention heads, and is not case sensitive. [4]

- RoBERTa-base (FacebookAI/roberta-base): The RoBERTa base model is pretrained on the union of five English language datasets weighing 160GB combined. This model has 125M parameters with 12 layers, 768 hidden layers, 12 attention heads, and is case sensitive. [5]
- DistilBERT-base (distilbert/distilbert-base-uncased): The DistilBERT base model is distilled from the BERT base uncased model. This model has 66M parameters, 6 layers, 768 hidden layers, 12 attention heads, and is not case sensitive. [6]
- BERTweet-base (vinai/bertweet-base): The BERTweet base model is pretrained on 850M English tweets weighing a combined 80GB, containing 845M tweets posted from 2012 to 2019 and 5M tweets related to the COVID-19 pandemic. The model has the same architecture as BERT base with 12 layers, 768 hidden layers, and 12 attention heads. [7]

In the preliminary phase of the experiment, each model's CLS, or Classification Start, token representation was used as input to a single linear classification layer to generate classification predictions. No layers were frozen, so all layers were fine-tuned on the TruthSeeker data. In the subsequent experimental phase, the linear classification layer was replaced with the DeepSeek LLM base model with 7B parameters, enabling projection to a higher-dimensional space. The DeepSeek LLM base model with 7B parameters is pretrained on a dataset of 2 trillion tokens in English and Chinese. [8] The LLM took the CLS embedding as input and produced final classification logits. With these model architectures loaded, the next step involved the training and evaluation of each classification pipeline.

4.2 Training Strategy

The training strategy for the models had the following consistent characteristics across both preliminary experiments and the final experiment with the LLM head:

- Learning rate: 4e-5
- Batch size: 32
- Gradient accumulation: 2
- Epochs: 10 with early stopping
- Optimizer: AdamW

During the preliminary experimental phase, two early stopping criteria were evaluated: F1 score and accuracy. Binary cross-entropy was used as the loss function, since the task was a binary classification. All training was conducted using the University of Miami's H100 GPUs on their IBM-built Pegasus 2 advanced computing cluster. [9] Throughout both phases, none of the layers of the models were frozen, allowing for fine-tuning of all layers on the TruthSeeker dataset. In the preliminary phase, the pre-trained transformer encodes the input text. Then the model processes the tokenized inputs and outputs contextual embeddings for each token in the sequence. From those embeddings, the CLS token representation is extracted, which is used to summarize the entire sequence. Then, in the preliminary phase, Hugging Face's AutoModelForSequenceClassification class from the transformers library was used to load the pretrained models and append a classification head for binary output.

In the final experimental phase, the linear classification head was replaced with an LLM-based module using DeepSeek's "deepseek-ai/deepseek-llm-7b-base" loaded from Hugging Face. While maintaining the same initial process as in the preliminary phase, this phase

then expands the CLS embedding to match the expected input size for the LLM in order to provide the input tokens for the LLM to produce binary classification logits.

4.3 Evaluation Metrics

Four primary metrics were used to evaluate model performance:

- **Accuracy**: Evaluates balanced overall performance in terms of the number of true positives and true negatives.
- **Precision**: Emphasizes the reduction of false positives in model output.
- **Recall**: Emphasizes the reduction of false negatives in model output.
- **F1 score**: Evaluates the balanced harmonic mean of precision and recall, reducing both false negatives and false positives.

In the context of this experiment, accuracy and F1 were ideal metrics for evaluation. Accuracy maximizes the correct classifications of tweets as real or fake. Since the real and fake classes were well-balanced in the input dataset, accuracy was a good option for early stopping and evaluation. F1 score minimizes incorrect classifications, evenly reducing real posts incorrectly flagged as fake and vice versa. This is significant in applications for monitoring social media posts, as hypersensitivity that leads to a high number of false positives could infringe on users' freedom of speech, while insufficient sensitivity leading to high false negatives allows fake posts to proliferate unfettered. In this case, F1 score helps to minimize the cost of incorrect classifications. With the contextual significance of reducing misclassifications and maximizing correct classifications, accuracy and F1 were the primary metrics of evaluation for model performance.

5. Experiments

5.1 Baseline Results (Linear Classifier)

In the preliminary phase of experimentation, each base model was fine-tuned using a single linear classification head. The models were evaluated using both F1-based and accuracy-based early stopping criteria to assess their relative performance across several key classification metrics.

Using early stopping on F1:

- BERT achieved the strongest overall performance across accuracy, F1, and recall.
- DistilBERT recorded the highest precision, indicating stronger resistance to false positives.
- BERTweet exhibited the fastest training time but lower performance metrics overall.

TruthSeeker Classification Task Results				
Early Stopping: F1				
	BERT	**BERTweet**	**RoBERTa**	**DistilBERT**
Accuracy	94.74%	94.54%	94.39%	94.69%
F1	94.92%	94.72%	94.59%	94.86%
Precision	94.79%	94.68%	94.44%	94.95%
Recall	95.05%	94.77%	94.73%	94.76%
Loss	0.2013	0.2038	0.2056	0.2068
Runtime (mins)	30	16	34	18
Epochs	6	7	7	7

Figure 3. Evaluation of model performance with early stopping based on F1 score.

Using early stopping on accuracy:

- BERT once again led in overall classification performance.
- BERTweet demonstrated the lowest training loss and fastest training runtime, making it a computationally efficient option.

TruthSeeker Classification Task Results				
Early Stopping: Accuracy				
	BERT	**BERTweet**	**RoBERTa**	**DistilBERT**
Accuracy	94.69%	94.54%	94.44%	94.58%
F1	94.87%	94.72%	94.63%	94.76%
Precision	94.73%	94.68%	94.52%	94.68%
Recall	95.02%	94.77%	94.74%	94.85%
Loss	0.2050	0.2038	0.2064	0.2079
Runtime (mins)	30	16	34	18
Epochs	6	7	7	7

Figure 4. Evaluation of model performance with early stopping based on accuracy.

Across both early stopping strategies, BERT consistently led, though the difference in performance between the models and the early stopping strategies were marginal. In both approaches, RoBERTa lagged behind in all metrics. Overall, results indicate that early stopping based on F1 yielded marginal but consistent improvements in classification quality. Consequently, F1-based early stopping was used in the final experimental phase.

5.2 Results with LLM-Based Classifier

In the final phase, the single linear classification head was replaced with the DeepSeek-LLM (7B parameters), allowing the CLS embedding to be projected into a higher-dimensional space for classification. Each BERT-based encoder produced CLS embeddings, which were passed to the LLM for prediction.

- All LLM-enhanced models stopped training after five epochs due to early stopping.
- DistilBERT+LLM marginally outperformed BERT+LLM in terms of accuracy and precision.
- BERT+LLM had the lowest loss and higher recall, suggesting better handling of false negatives.
- DistilBERT+LLM recorded the fastest runtime of 17 minutes, outperforming other LLM-enhanced pipelines.

TruthSeeker Classification Task Results				
LLM Head, Early Stopping: F1				
	BERT	**BERTweet**	**RoBERTa**	**DistilBERT**
Accuracy	94.71%	94.70%	94.51%	94.73%
F1	94.90%	94.88%	94.68%	94.90%
Precision	94.57%	94.79%	94.86%	94.95%
Recall	95.23%	94.96%	94.50%	94.85%
Loss	0.2012	0.2099	0.2146	0.2084
Runtime (mins)	32	22	34	17
Epochs	5	5	5	5

Figure 5. Evaluation of model performance with the LLM classification head and early stopping based on F1 score.

6. Discussion

The experimental results affirm that BERT-based transformer models are effective tools for automated fake news detection in cybersecurity contexts. Among the base architectures, BERT consistently outperformed other models across multiple metrics, aligning with prior research indicating its reliability and robust contextual understanding. DistilBERT emerged as a compelling lightweight alternative, particularly when enhanced with an LLM-based classifier.

Replacing the traditional linear classification head with an LLM provided modest performance improvements, with DistilBERT+LLM outperforming all other pipelines in both runtime and precision. This suggests that for applications where processing speed and false positive mitigation are essential, such as real-time content moderation, LLM-enhanced lightweight models may offer a strategic advantage. DistilBERT is particularly effective for this use case, as its performance was enhanced and runtime was reduced when combined with the LLM classifier, while other models had increased runtimes and similar performance.

However, the use of LLMs in classification pipelines introduces increased computational demands, which could hinder real-time deployment without specialized hardware and any potential for scaling this approach. Furthermore, BERTweet's underperformance challenges the assumption that domain-specific pretraining guarantees superior results. This outcome reinforces the need for comprehensive empirical testing when selecting models for deployment in cybersecurity infrastructures.

Lastly, while the TruthSeeker dataset provided a valuable benchmark, its limited documentation around preprocessing decisions poses potential risks to reproducibility and trustworthiness—two critical concerns in cybersecurity research and policy.

7. Conclusion

This study explored the use of BERT-based transformer models for automated fake news detection and introduced a novel architecture stacking these models with a large language model (LLM) classification head. The research aimed to determine optimal model architecture, early stopping strategy, and the efficacy of high-dimensional projection for classification.

Key takeaways include:

- Best base model: BERT-base achieved the most reliable performance with a linear classifier.
- Best LLM-enhanced model: DistilBERT+LLM achieved the best balance of accuracy, precision, and training time.
- Optimal early stopping: F1-based early stopping marginally improved classification performance.
- LLM integration: Provided limited but measurable gains, with potential for real-time efficiency in lighter models.

In the realm of cybersecurity, where the detection and mitigation of disinformation campaigns are increasingly urgent, these findings offer evidence that hybrid transformer-LLM architectures can be adapted to scalable, automated pipelines. Future work should investigate explicit hyperparameter tuning, dataset robustness, and model explainability to further enhance trust and transparency in AI-driven disinformation detection systems.

References

[1] J. Y. Khan, M. T. I. Khondaker, S. Afroz, G. Uddin, and A. Iqbal, “A benchmark study of machine learning models for online fake news detection,” *Machine Learning with Applications*, vol. 4, p. 100032, 2021. [Online]. Available: https://doi.org/10.1016/j.mlwa.2021.100032

[2] R. Kozik, A. Pawlicka, M. Pawlicki, M. Choraś, W. Mazurczyk, and K. Cabaj, “A Meta-Analysis of State-of-the-Art Automated Fake News Detection Methods,” *IEEE*

Transactions on Computational Social Systems, vol. 11, no. 4, pp. 5219–5229, Aug. 2024, doi: 10.1109/TCSS.2023.3296627.

[3] S. Dadkhah, X. Zhang, A. G. Weismann, A. Firouzi, and A. A. Ghorbani, "The Largest Social Media Ground-Truth Dataset for Real/Fake Content: TruthSeeker," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 3, pp. 3376–3390, Jun. 2024, doi: 10.1109/TCSS.2023.3322303.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint*, arXiv:1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[5] Y. Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv preprint*, arXiv:1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[6] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv preprint*, arXiv:1910.01108, 2019.

[7] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "BERTweet: A pre-trained language model for English Tweets," in *Proc. 2020 Conf. on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 9–14, 2020.

[8] DeepSeek-AI, "DeepSeek LLM: Scaling Open-Source Language Models with Longtermism," *arXiv preprint*, arXiv:2401.02954, 2024. [Online]. Available: https://huggingface.co/deepseek-ai/deepseek-llm-7b-base

[9] University of Miami Institute for Data Science and Computing, “Pegasus,” [Online]. Available: https://idsc.miami.edu/pegasus/

[10] C. R. Harris et al., “Array programming with NumPy,” *Nature*, vol. 585, pp. 357–362, 2020, doi: 10.1038/s41586-020-2649-2.

[11] The pandas development team, “pandas-dev/pandas: Pandas,” *Zenodo*, Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[12] T. Wolf et al., “HuggingFace’s Transformers: State-of-the-art Natural Language Processing,” *arXiv preprint*, arXiv:1910.03771, 2020. [Online]. Available: https://arxiv.org/abs/1910.03771

Appendix

Full training parameters:

```
def train_model(model_name, dataset):
        tokenizer_kwargs["normalization"] = True

    # Save tokenizer for later use
    tokenizer = AutoTokenizer.from_pretrained(MODEL_MAP[model_name], **tokenizer_kwargs)
    tokenizer.save_pretrained(os.path.join(model_dir, "tokenizer"))
    logger.info(f"Tokenizer saved to {model_dir}/tokenizer")

    # CRITICAL: ADDING CLASSIFICATION HEAD HERE
    model = AutoModelForSequenceClassification.from_pretrained(MODEL_MAP[model_name], num_labels=2)

    # Tokenization
    train_dataset = tokenize_data(train_dataset, tokenizer, model_name)
    val_dataset = tokenize_data(val_dataset, tokenizer, model_name)
    test_dataset = tokenize_data(test_dataset, tokenizer, model_name)

    # Collator for dynamic padding
    data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

    # Define training arguments
    training_args = TrainingArguments(
        output_dir=os.path.join(model_dir, "checkpoints"),
        eval_strategy="epoch",
        save_strategy="epoch",
        learning_rate=4e-5,
        per_device_train_batch_size=32,
        per_device_eval_batch_size=32,
        gradient_accumulation_steps=2,
        num_train_epochs=10,
        weight_decay=0.01,
        logging_dir=os.path.join(model_dir, "logs"),
        report_to="none",  # Disable logging to W&B by default
        logging_steps=10,
        load_best_model_at_end=True,
        metric_for_best_model="accuracy",  # set metric for early stopping
        greater_is_better=True,  # set to True for accuracy or F1
        adam_beta1=0.9,
        adam_beta2=0.999,
        adam_epsilon=1e-8,
        run_name=f"{model_name}_run",  # custom run name
        # Additional arguments for better logging
        logging_first_step=True,
        save_total_limit=3,  # only keep the 3 best checkpoints
        push_to_hub=False
        # push_to_hub=True,
        # hub_model_id=f"wildgeese25/{model_name.replace('_','-')}-fake-news-detector-2",
        # hub_strategy="end"
    )
```

```
# Initialize custom callback
metric_callback = MetricCallback(model_name, model_dir, patience=3, threshold=0.0)

# Initialize trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    processing_class=tokenizer,  # Using processing_class instead of deprecated tokenizer parameter
    callbacks=[metric_callback],
    data_collator=data_collator,
    compute_metrics=compute_metrics
)
```

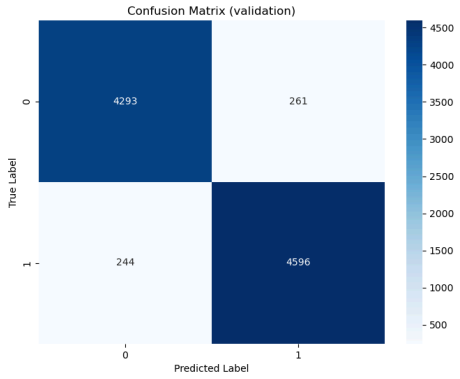
StackedTransformerLLM class:

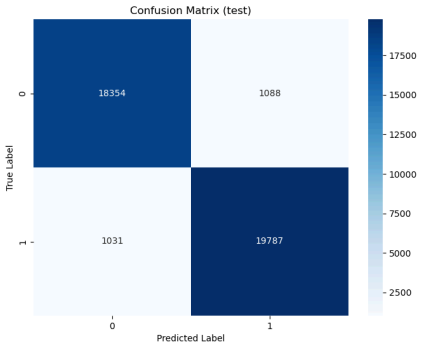
```python
class StackedTransformerLLM(nn.Module):
    def __init__(self, base_model_name, llm_model_name, num_labels=2):
        super(StackedTransformerLLM, self).__init__()

        # Load pre-trained base model (e.g., BERT)
        self.base_model = AutoModel.from_pretrained(base_model_name)

        # Load small LLM for classification
        self.llm_model = AutoModelForSequenceClassification.from_pretrained(
            llm_model_name,
            num_labels=num_labels  # final output classes
        )

    def forward(self, input_ids, attention_mask=None, token_type_ids=None, labels=None):
        # Process different input parameters based on model type
        base_inputs = {
            'input_ids': input_ids,
            'attention_mask': attention_mask
        }

        # Add token_type_ids if provided (needed for BERT but not for all models)
        if token_type_ids is not None:
            base_inputs['token_type_ids'] = token_type_ids

        # 1. Pass through base transformer
        base_outputs = self.base_model(**base_inputs)
        hidden_states = base_outputs.last_hidden_state  # shape: (batch_size, seq_len, hidden_d

        # 2. Pool the [CLS] token (first token)
        cls_embedding = hidden_states[:, 0, :]  # shape: (batch_size, hidden_dim)

        # 3. Expand to simulate sequence input for LLM
        cls_embedding_expanded = cls_embedding.unsqueeze(1)  # shape: (batch_size, 1, hidden_di

        # 4. Pass through LLM model (using inputs_embeds instead of input_ids)
        outputs = self.llm_model(inputs_embeds=cls_embedding_expanded, labels=labels)

        return outputs
```

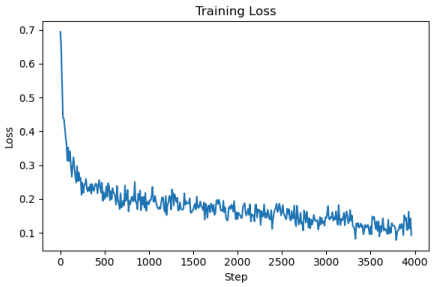
Model Evaluation Visualizations:

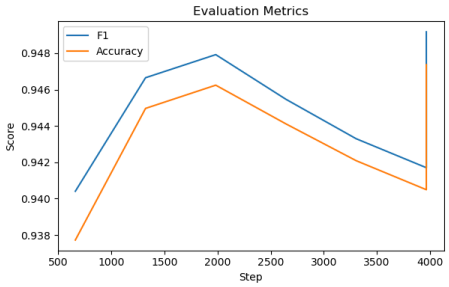
Initial Phase: Early Stopping with F1

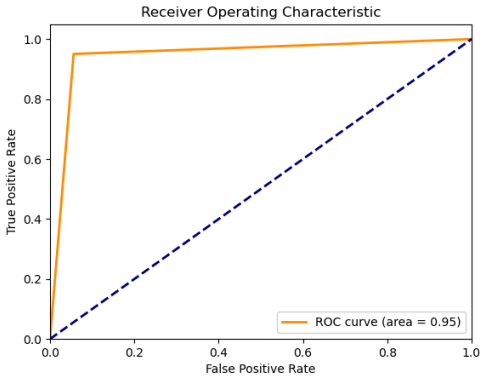
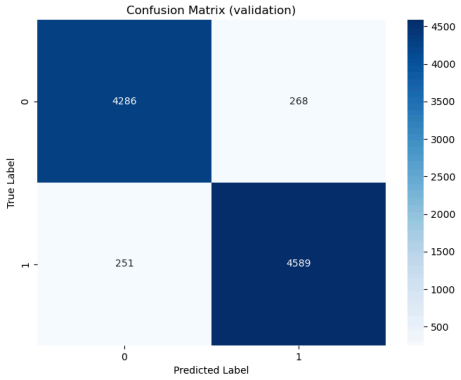
BERT

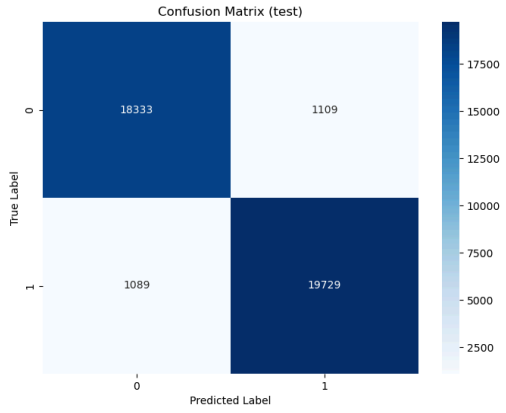

Confusion Matrix (validation)
4293
261
244
4596
True Label
Predicted Label
0
1
4500
4000
3500
3000
2500
2000
1500
1000
500

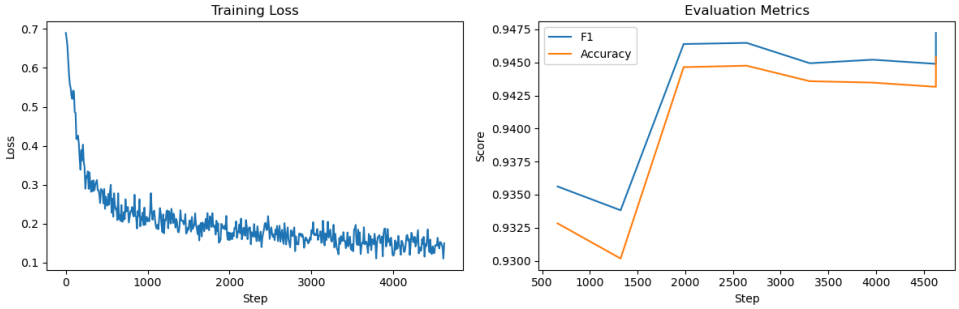

Confusion Matrix (test)
18354
1088
1031
19787
True Label
Predicted Label
0
1
17500
15000
12500
10000
7500
5000
2500

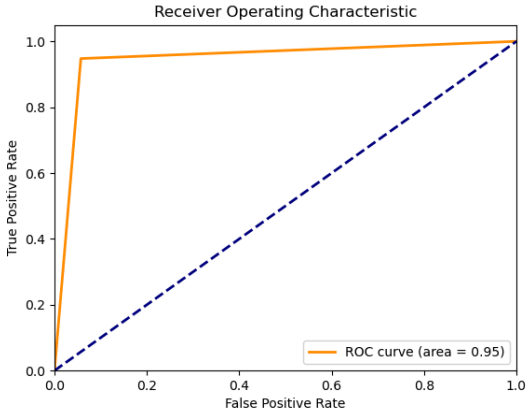

Training Loss
Loss
Step
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
2000
2500
3000
3500
4000


Evaluation Metrics
F1
Accuracy
Score
Step
0.948
0.946
0.944
0.942
0.940
0.938
500
1000
1500
2000
2500
3000
3500
4000

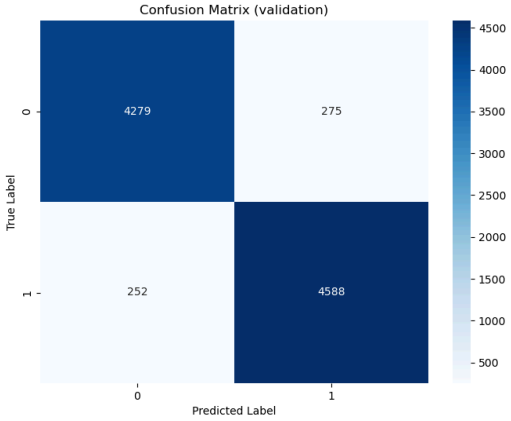
<table>
<tr><td></td><td> 
</td></tr>
<tr><td>BERTweet</td><td> 

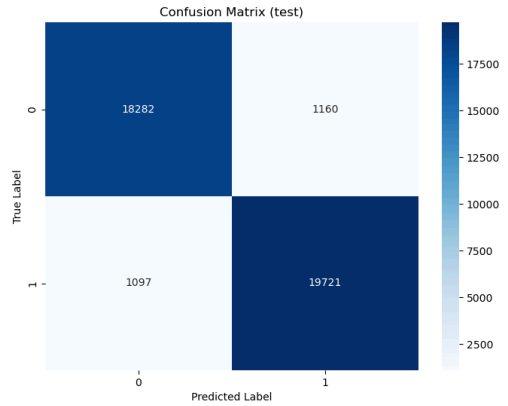

</td></tr>
</table>

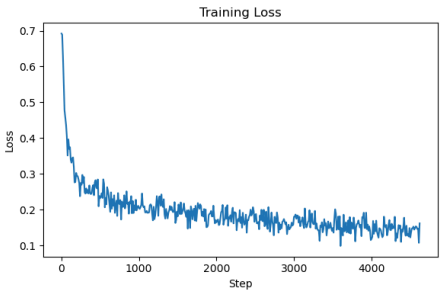

Training Loss
Loss
Step
Evaluation Metrics
F1
Accuracy
Score
Step

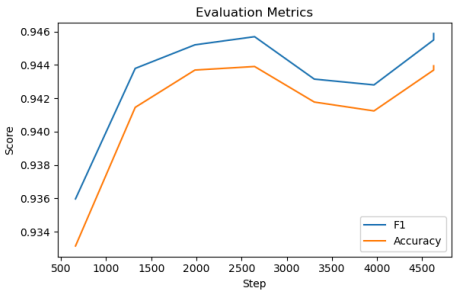

Receiver Operating Characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.95)

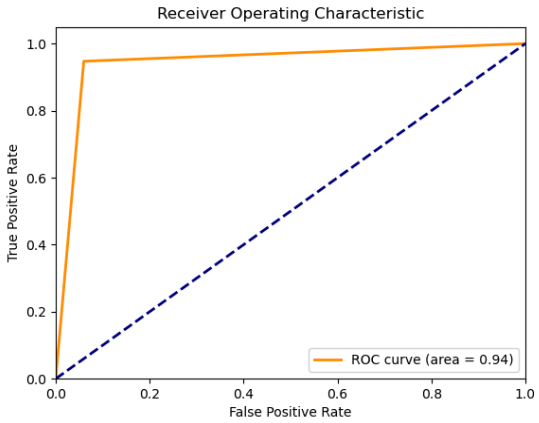
RoBERTa


Confusion Matrix (validation)
4279
275
252
4588
True Label
Predicted Label

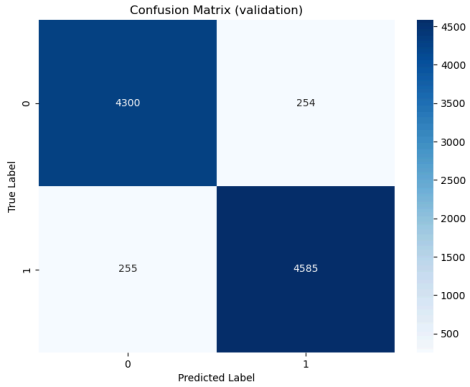

Confusion Matrix (test)
18282
1160
1097
19721
True Label
Predicted Label
0
1
17500
15000
12500
10000
7500
5000
2500

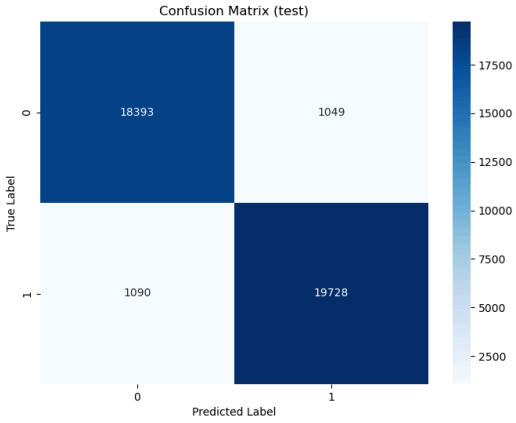

Training Loss
Loss
Step
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1000
2000
3000
4000

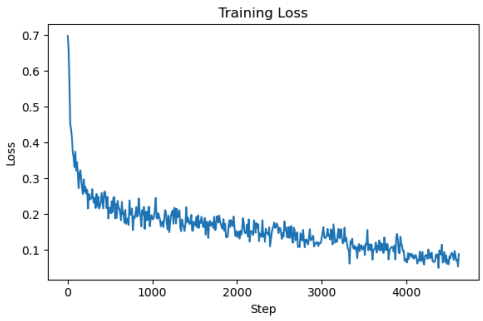

Evaluation Metrics
Score
Step
F1
Accuracy
0.946
0.944
0.942
0.940
0.938
0.936
0.934
500
1000
1500
2000
2500
3000
3500
4000
4500

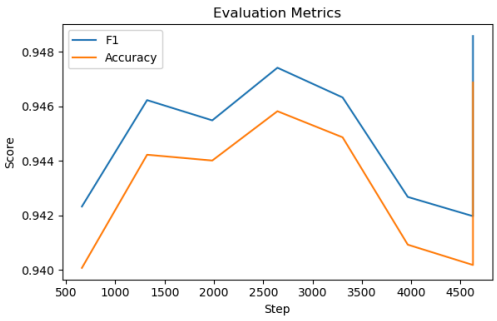

Receiver Operating Characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.94)
0.0
0.2
0.4
0.6
0.8
1.0

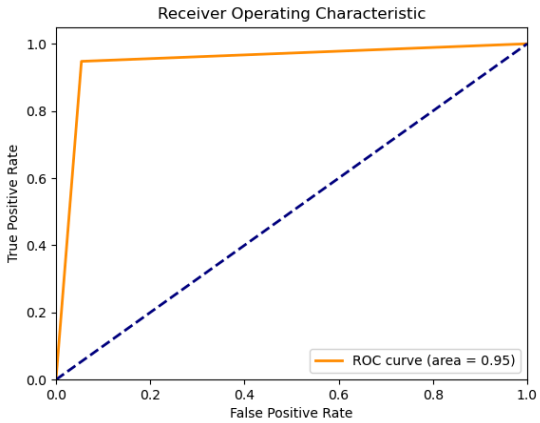
DistilBERT

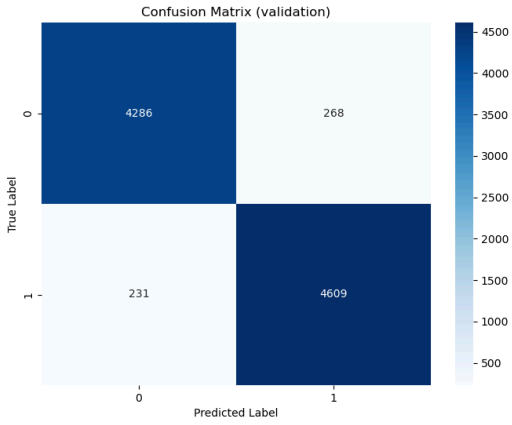

Confusion Matrix (validation)
4300
254
255
4585
True Label
Predicted Label
0
1
4500
4000
3500
3000
2500
2000
1500
1000
500

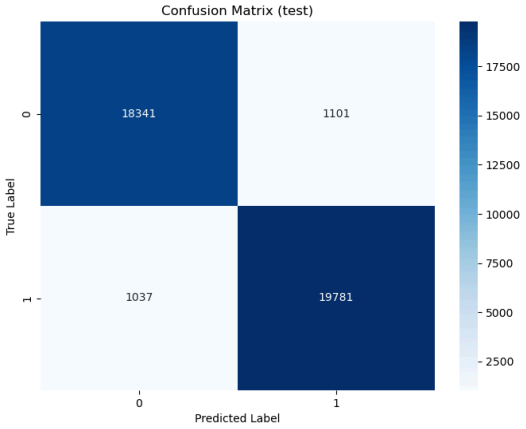

Confusion Matrix (test)
18393
1049
1090
19728
True Label
Predicted Label
0
1
17500
15000
12500
10000
7500
5000
2500

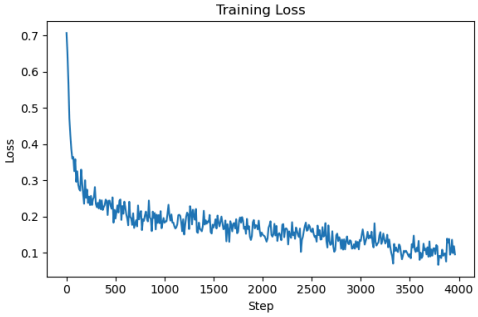

Training Loss
Loss
Step
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1000
2000
3000
4000

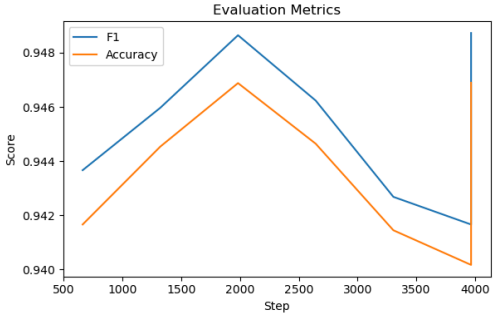

Evaluation Metrics
F1
Accuracy
Score
Step
0.948
0.946
0.944
0.942
0.940
500
1000
1500
2000
2500
3000
3500
4000
4500

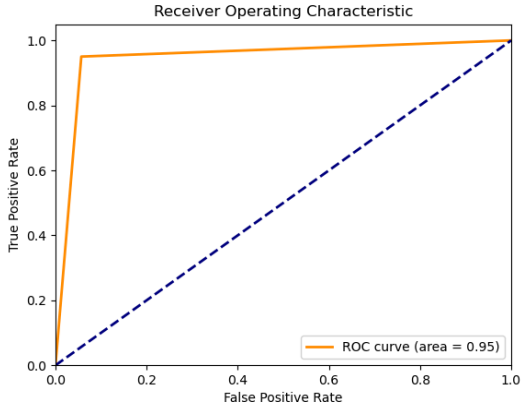
	

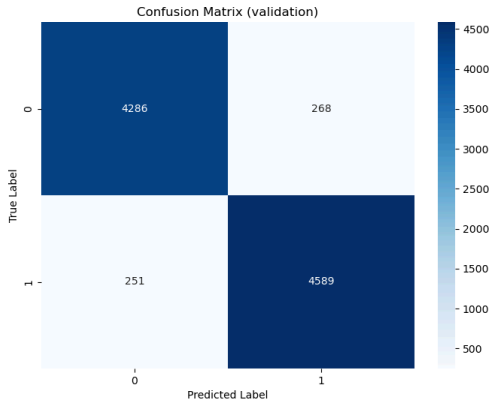
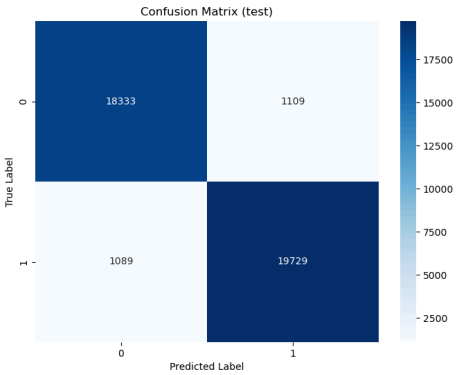
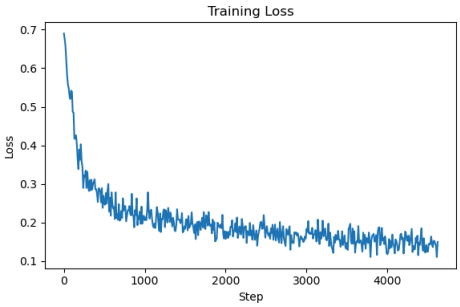
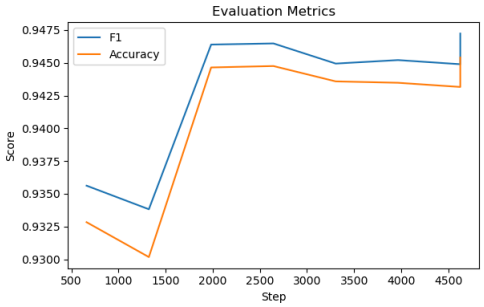
Initial Phase: Early Stopping with Accuracy	
BERT	

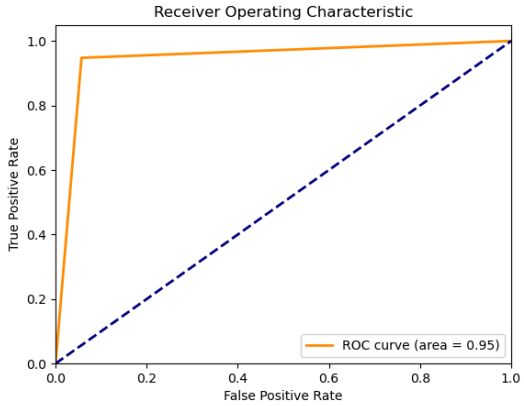
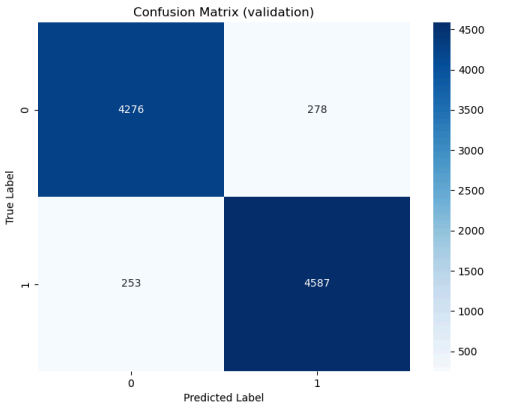
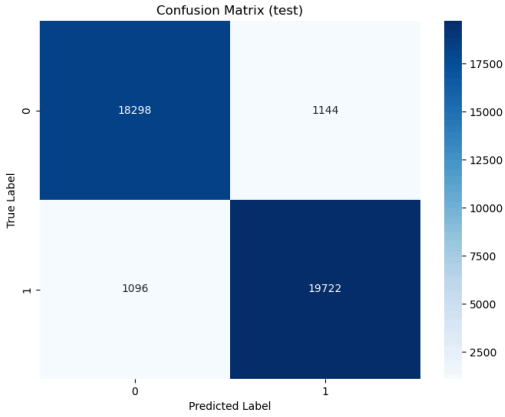

Confusion Matrix (test)
18341
1101
1037
19781
True Label
Predicted Label
0
1
17500
15000
12500
10000
7500
5000
2500

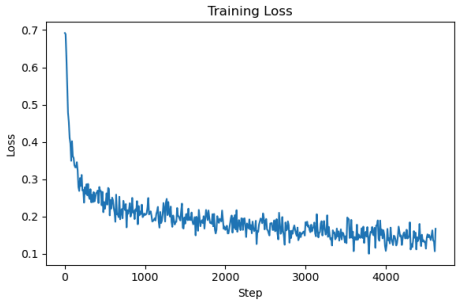

Training Loss
Loss
Step
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
2000
2500
3000
3500
4000

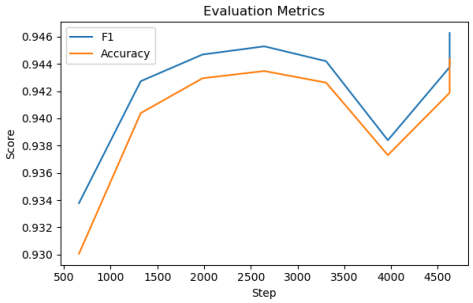

Evaluation Metrics
F1
Accuracy
Score
Step
0.948
0.946
0.944
0.942
0.940
500
1000
1500
2000
2500
3000
3500
4000

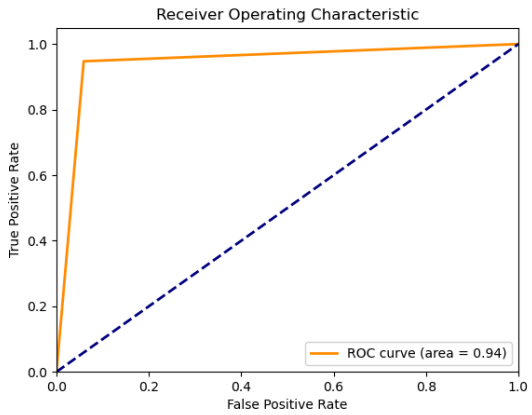

Receiver Operating Characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.95)
0.0
0.2
0.4
0.6
0.8
1.0

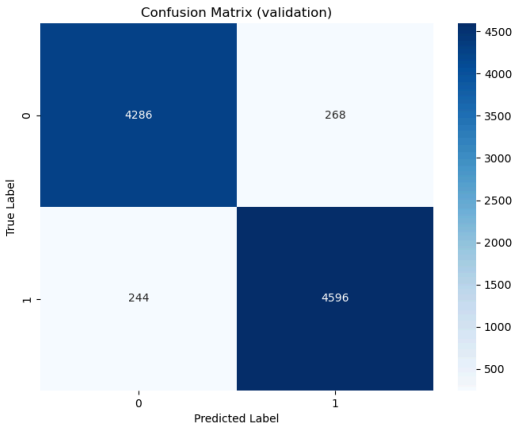
BERTweet	   

	 Receiver Operating Characteristic True Positive Rate False Positive Rate ROC curve (area = 0.95) 0.0 0.2 0.4 0.6 0.8 1.0
RoBERTa	 Confusion Matrix (validation) 4276 278 253 4587 True Label Predicted Label 0 1 4500 4000 3500 3000 2500 2000 1500 1000 500  Confusion Matrix (test) 18298 1144 1096 19722 True Label Predicted Label 0 1 17500 15000 12500 10000 7500 5000 2500

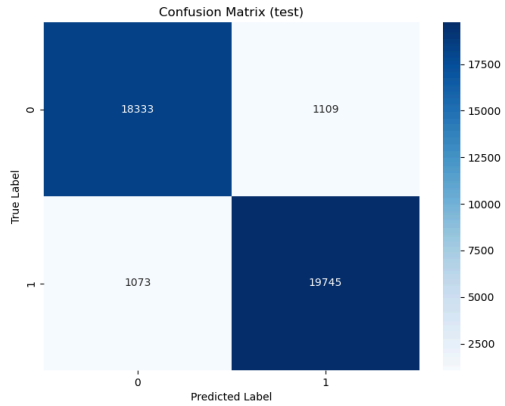
<table>
<tr>
<td></td>
<td>


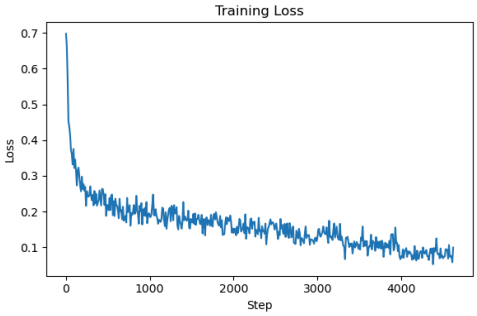


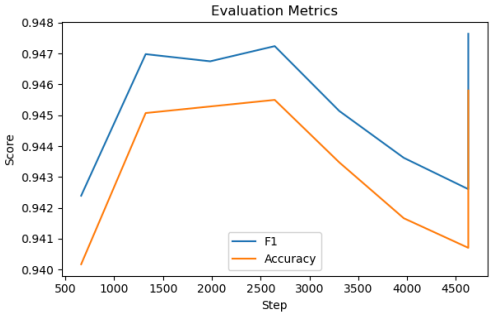


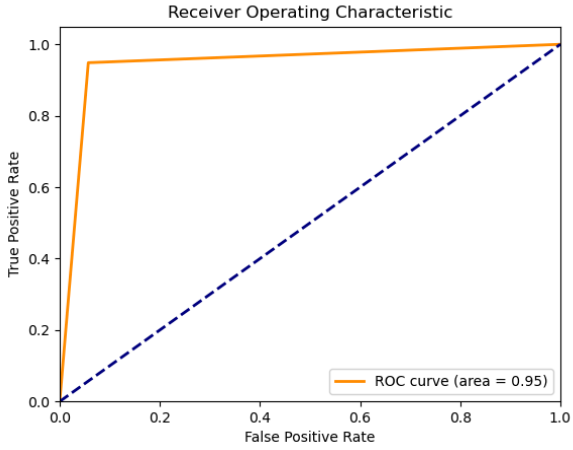
</td>
</tr>
<tr>
<td>DistilBERT</td>
<td>


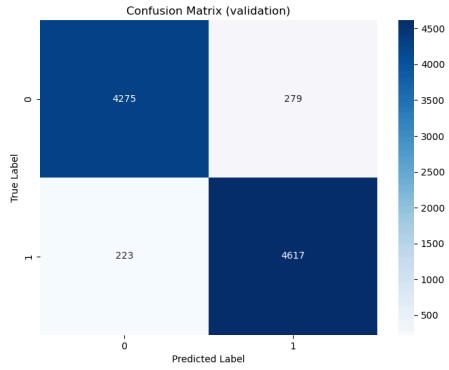
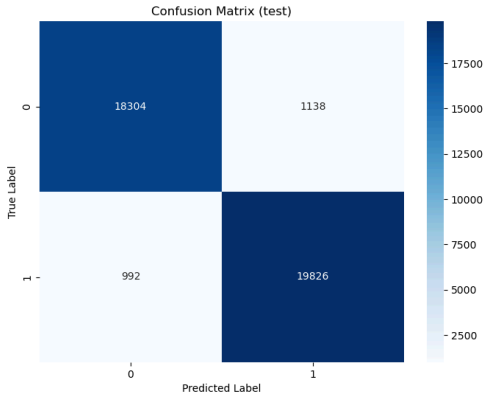
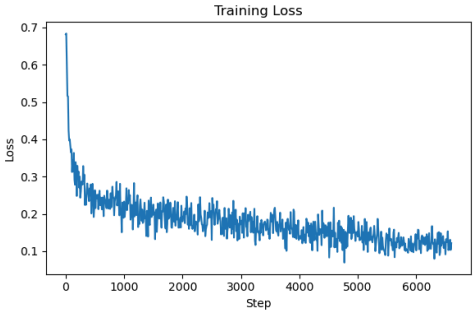
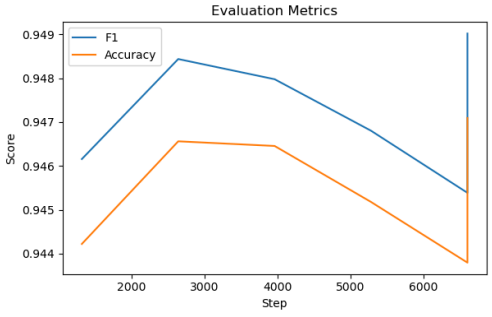
</td>
</tr>
</table>

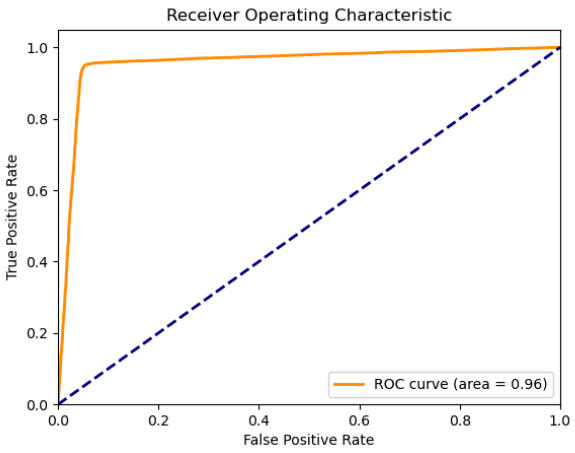
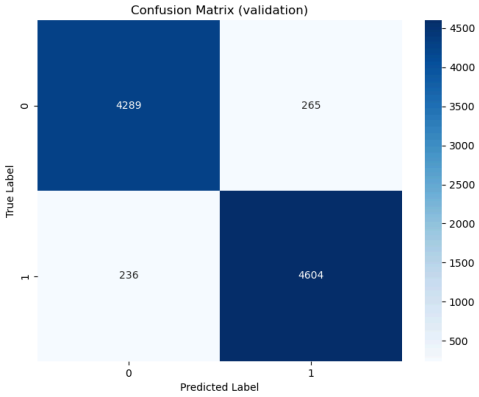
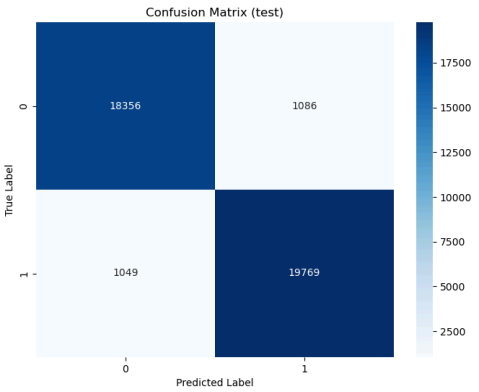

Confusion Matrix (test)
18333
1109
1073
19745
True Label
Predicted Label
0
1
17500
15000
12500
10000
7500
5000
2500

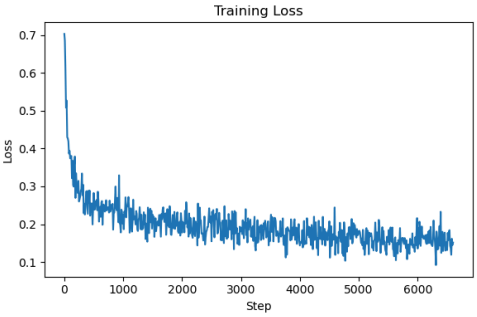

Training Loss
Loss
Step
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1000
2000
3000
4000

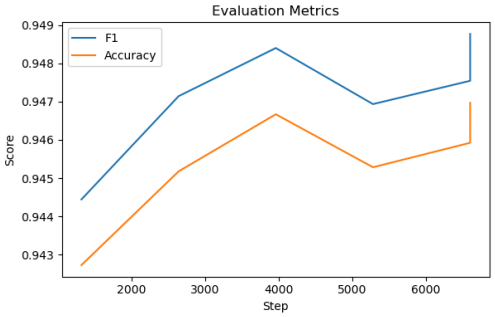

Evaluation Metrics
Score
Step
F1
Accuracy
0.948
0.947
0.946
0.945
0.944
0.943
0.942
0.941
0.940
500
1000
1500
2000
2500
3000
3500
4000
4500

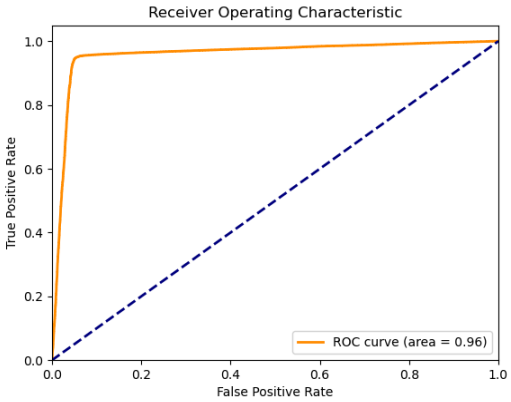

Receiver Operating Characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.95)
0.0
0.2
0.4
0.6
0.8
1.0

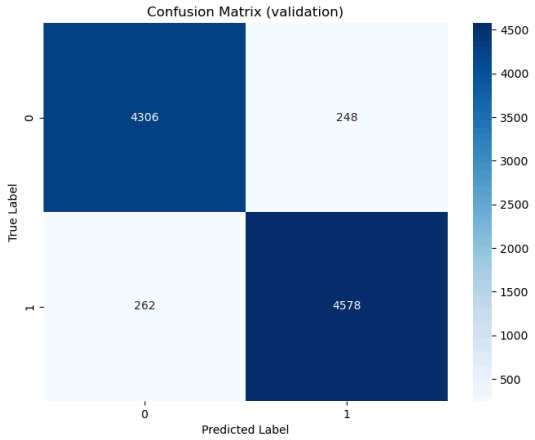
Final Phase: LLM Head, Early Stopping with F1	
BERT	   

	 Receiver Operating Characteristic True Positive Rate False Positive Rate ROC curve (area = 0.96) 0.0 0.2 0.4 0.6 0.8 1.0
BERTweet	 Confusion Matrix (validation) 4289 265 236 4604 True Label Predicted Label 0 1 4500 4000 3500 3000 2500 2000 1500 1000 500  Confusion Matrix (test) 18356 1086 1049 19769 True Label Predicted Label 0 1 17500 15000 12500 10000 7500 5000 2500

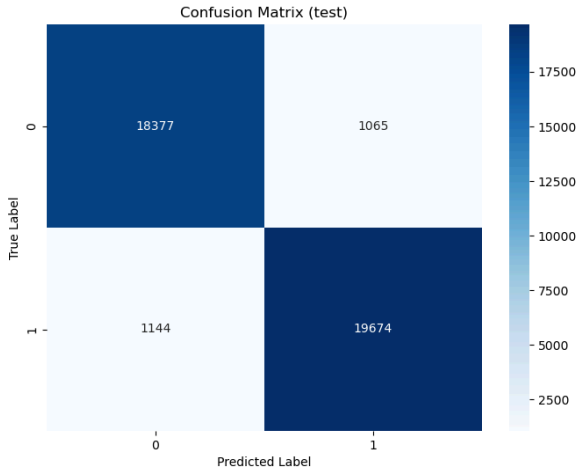
<table>
<tr>
<td></td>
<td>


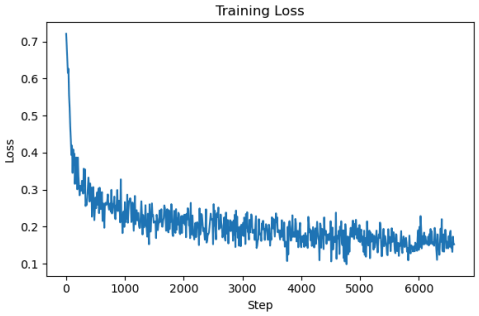


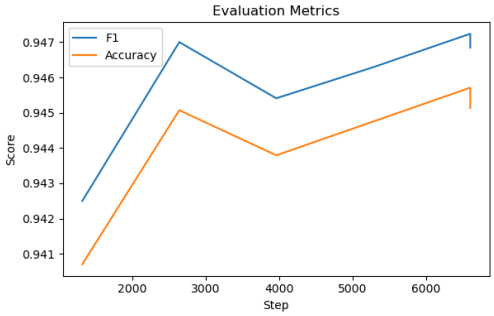


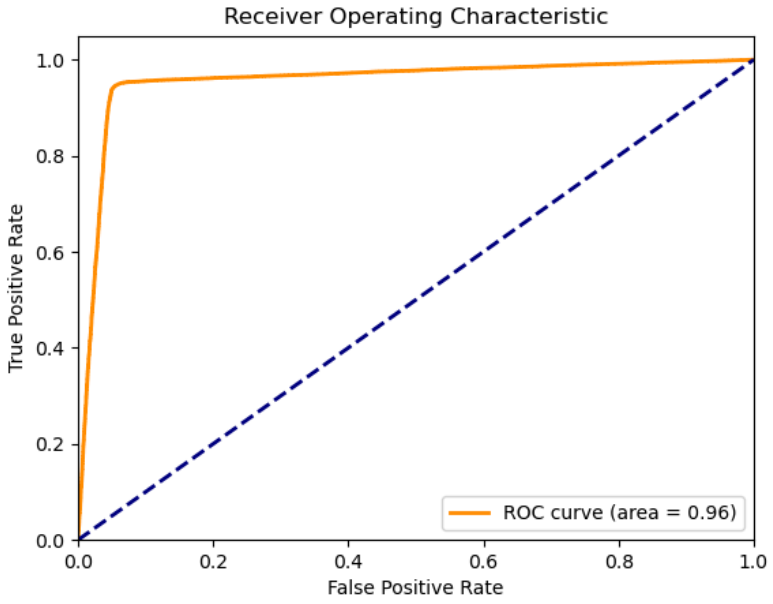
</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>


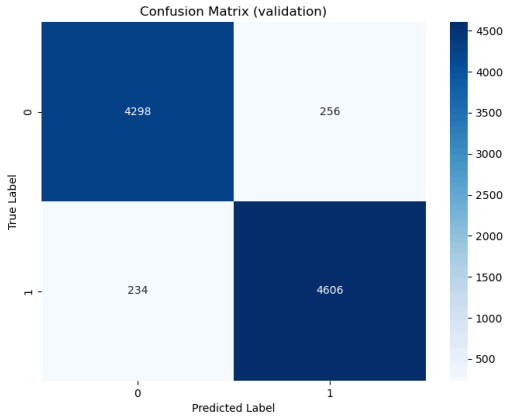
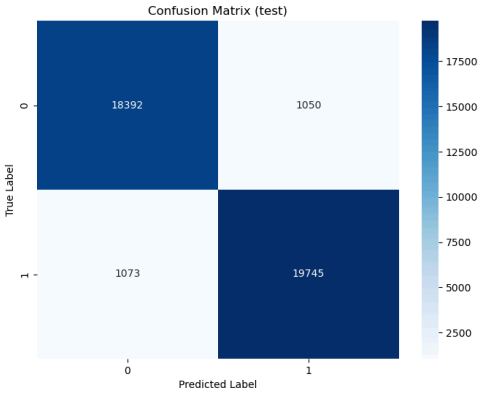
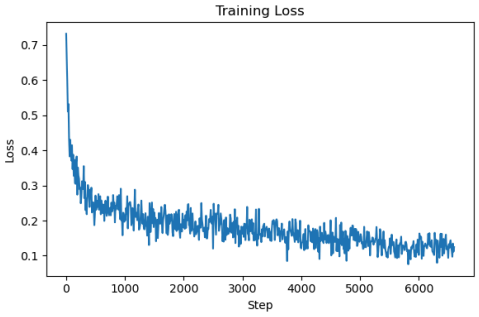
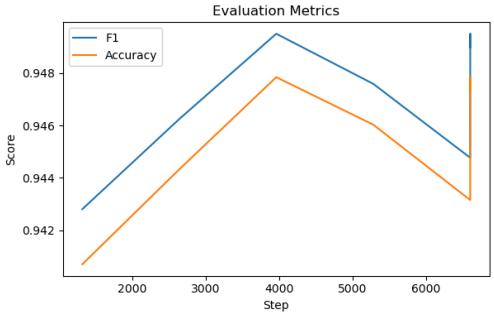
</td>
</tr>
</table>

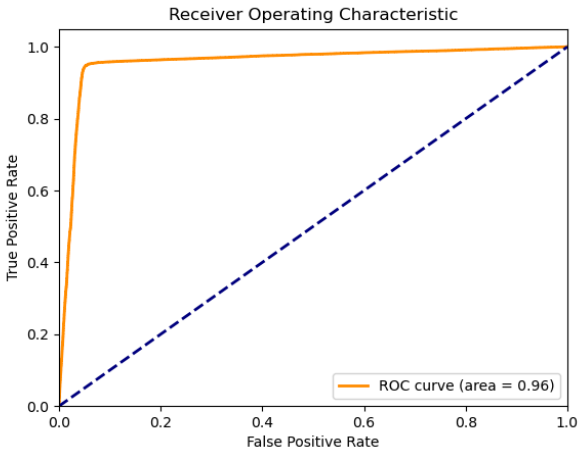

Confusion Matrix (test)
18377
1065
1144
19674
True Label
Predicted Label
0
1
17500
15000
12500
10000
7500
5000
2500


Training Loss
Loss
Step
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1000
2000
3000
4000
5000
6000


Evaluation Metrics
F1
Accuracy
Score
Step
0.947
0.946
0.945
0.944
0.943
0.942
0.941
2000
3000
4000
5000
6000


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.96)
0.0
0.2
0.4
0.6
0.8
1.0

DistilBERT	   


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.96)