Detecting Sarcasm in Text Data From Twitter

Eshan Bhatnagar, Mahmoud Ghanem, Michael Kalish
W 266: Natural Language Processing
UC Berkeley School of Information
{eshan_bhatnagar, michael.kalish, mghanem}@ischool.berkeley.edu

Abstract
Understanding sarcasm is an important skill in communicating with peers. Detecting sarcasm in text on social media or other online platforms can be a challenge without the cues that communicating in person provides. This project explores the use of three BERT based models (BERT, BERTweet, and ALBERT) in classifying text data taken from Twitter as sarcastic or not. These models were enhanced by concatenating the embeddings from the BERT models with Word2Vec embeddings. The high precision obtained from the BERTweet model with concatenated Word2Vec embeddings suggests a promising approach and future opportunities for refinements are recommended.

1 Introduction

Sarcasm is a prevalent aspect of language and how we communicate with others. It serves as a form of expression characterized by irony and wit and is used to convey complex emotions, to critique societal norms, or to simply engage in playful and humorous banter. In the age of social media, sarcasm is commonly expressed in online interactions through written text and the ability to interpret sarcasm is an essential skill for effectively engaging with peers and deciphering nuanced messages. Below are the benefits and challenges of creating a model that can effectively detect sarcasm.

Benefits

- Help neurodivergent people better identify sarcastic notes and cues to better navigate online social environments
- Help detect sarcasm when interpreting customer feedback for business purposes
- Help disambiguate sarcastic speech that may be wrongfully flagged in public forums from actually harmful speech

Challenges

- Sarcastic language can model serious language with sometimes subtle differences.
- Tweets are constrained by their character length, which challenges users to be creative in their post structure and content. The result is a fairly unique style of writing that can differ from the corpora used to generate the default embeddings for traditional large language models.
- Hyperbolic language can be sarcastic, but this is not necessarily true for all instances.
- Detecting sarcasm in text can be more challenging without cues such as facial expressions, body language, or changes in tone that speaking in person provides. Broader context of the overall statement is more important for evaluating sarcasm.

Why a new approach?

Creating custom embeddings is important to align with the form and character of language in a tweet to maximize model performance. This paper explores the effectiveness of pairing Hugging Face models (BERT, ALBERT, and BERTweet) with custom embeddings to identify sarcasm in text data taken from Twitter.

2 Background

2.1 Dataset Used

The datasets derive from a collection used for the purposes of the following publication: UTNLP at SemEval-2022 Task 6: A Comparative Analysis of Sarcasm Detection using generative-based and mutation-based data augmentation [1]. The data used derives from the article's dataset mapping for train3 and train5, which are **"**Twitter News Dataset" and "SPIRS", respectively. A sample of 5,000 tweets were shuffled and used for the train set. 1,200 were used for the validation set and 3,000 were used for the test set. For each set, the data was balanced by label (class_1 = sarcastic, class_0 == not sarcastic).

2.2 Related Work

In the paper *Sarcasm Detection in a Disaster Context*, BERTweet models were used to detect sarcasm in tweets and the highest performing model generated an F1 score of 0.70 on the dataset [2]. BERTweet is a pre-trained model that has the same architecture as BERT-base, but is catered toward English tweets, which is what our dataset is composed

of [3]. This inspired us to implement it and see if we could get better performance results on our dataset after fine-tuning the model.

In the paper *Combining BERT with Static Word Embeddings for Categorizing Social Media*, an architecture is described where static word embeddings are passed through a CNN, and after global max pooling, are concatenated to a pooled BERT layer to effectively combine representations from the BERT and static word vectors at a sentence level. This model was then used for multiple social media classification tasks, including irony detection, where its highest performance generated an F1 score of 0.684 [4]. We decided to adopt a similar architecture to see how concatenating custom Word2Vec embeddings could enhance the performance of the baseline BERT models.

ALBERT (A Lite BERT) was picked as a third model to experiment with since it has a similar architecture to BERT, but improves on performance by increasing training speed and reducing memory consumption. ALBERT decomposes the large vocabulary embedding matrix into two smaller matrices, separating the size of the hidden layers from the size of the vocabulary embeddings, allowing the hidden size to grow without increasing the parameter size of the vocabulary embeddings. ALBERT also uses cross-layer parameter sharing, which prevents parameters from growing with the depth of the network. With these two parameter reduction techniques, ALBERT can be trained faster with fewer parameters with stabilization to improve generalization [5].

3 Methods

3.1 Baseline Models

We employed three well-established baseline models for BERT, BERTweet, and ALBERT respectively. For these models, custom Word2Vec embeddings were not included, and each model used embeddings generated by the model directly. We extracted the pooled output as fixed-size representation for each input sequence, derived from the hidden states to capture the essence of the input content for classification which were then passed through a classification layer for the binary classification output.

3.1.1 Challenges of Static Embeddings

During our early experiments we used static embeddings to reduce our model consumption of computational units. However, this approach presented the following issues:

- BERT's architecture allows it to dynamically adjust embeddings based on the surrounding text, providing a rich, context-aware representation of language [6]. In word2vec embeddings a fixed representation lacks the ability to adapt to context beyond the sentence level. When we used static BERT embeddings, the model's ability to fine-tune the contextual nuances was decreased, resulting in a loss of the nuanced detection capabilities essential for our approach [7].

To overcome these challenges, we revised our experiment's approach to ensure that both types of embeddings contribute effectively to the model without undermining each other's strengths. We adopted a methodology where we used dynamic embeddings directly from BERT models and both were processed through separate pathways before being concatenated. This strategy allowed the integration of both embedding type's representations.

3.2 Custom Models

To enhance the detection capabilities of our baseline models we developed six additional models that were created to improve upon the baseline models. All of these models include the custom Word2Vec embeddings integrated with the transformer architectures. This integration aims to leverage the localized context-specific information captured by Word2Vec with the deep contextual insights provided by transformers. In order to create these embeddings, we first cleaned the tweets by removing URLS, user mentions, and excessive whitespace. We then trained the Word2Vec with the tweets and converted the tweets to a sequence of indices. The sequences were then padded and projected to a linear transformation layer.

3.2.1 Model architecture

The following table summarizes the architecture of our custom models:

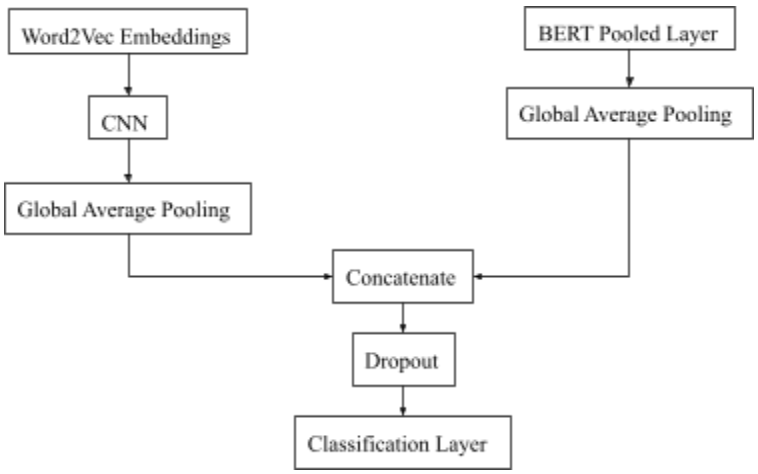


Fig 1. Architecture of concatenating BERT and Word2Vec embeddings

Experiment	Embedding	Models				Parameters
Model name	#	w2v	Bert	BertTweet	Albert	Base	Base + Freeze + Attention
BERT	1		x				
BERTweet	2			x			
ALBERT	3				x		
CNNForWord2VecBERT	4	x	x			x	
CNNForWord2VecBERTFT	5	x	x				x
CNNForWord2VecBERTweet	6	x		x		x	
CNNForWord2VecBERTweetF	7	x		x			x
CNNForWord2VecALBERT	8	x			x	x	
CNNForWord2VecALBERTFT	9	x			x		x

Fig 2. Structure of all nine models

3.2.2 Feed forward method: inputs and design

The idea of concatenating the Word2Vec embeddings to the BERT embeddings was to enhance representation of the text data and add more information to the model with increased dimensionality, as Word2Vec can capture certain linguistic patterns and semantic relationships that BERT models cannot [7]. Word2Vec embeddings provide valuable insights into the word level that are foundational to understanding language, while BERT focuses on context meanings, adjusting its understanding of words based on their surrounding text help where words have multiple meanings depending on context. The approach of concatenating Word2Vec embeddings with BERT embeddings seeks to utilize the strengths of both approaches. We aimed to leverage Word2Vec's ability in capturing deep word understanding with BERT's dynamic word contextual understanding to enhance the model's ability to detect subtleties of language in sarcasm on both word and sentence level, giving the model better performance at interpreting the multifaceted nature of human communication found in social media.

However, the Word2Vec vectors could not simply be concatenated to the BERT vectors due to differences in the tokenization strategy. Word2Vec tokenizes by word, whereas BERT tokenization may split words into multiple word-piece tokens. This could cause problems in aligning the tokens during concatenation. Therefore, we followed a similar strategy used by the *Combining BERT with Static Word Embeddings for Categorizing Social Media* paper and concatenated the embeddings after pooling [4].

The dimensionality of the Word2Vec embeddings was set to 300. Word2Vec embeddings were passed through the convolutional neural network and global average pooling was applied to get pooled vectors. Global average pooling was used for BERT embeddings. The pooled embeddings were then concatenated and subject to a dropout layer before passing through the classification layer. The total dimensionality of the concatenated embeddings was 1068. This architecture was adopted for all three BERT models used, and the models were titled CNNForWord2VecBERT, CNNForWord2VecBERTweet, and CNNForWord2VecALBERT.

Additional Components	Output Layer
CNN, Global Avg Pooling	Concatenation, FC

Fig 3. BERT models with concatenated Word2Vec embeddings

3.2 Fine-tuning

Further fine-tuning was done to these three models to increase performance even more. While processing the BERT embeddings prior to concatenating them with the pooled Word2Vec embeddings, an attention mask was applied to all non-padded tokens. Since understanding the context of tweets would be especially important in determining whether they were sarcastic, applying attention to specific tokens would be especially useful in enhancing the model to do so. Additionally, we froze the BERT layer during the forward pass to keep the weights and biases constant during backpropagation. Freezing the BERT layer aimed to prevent overfitting the model during the training and focus on fine-tuning the Word2Vec embeddings and maintain the BERT's pre-trained, context-aware embeddings which also could reduce the computational load. These fine-tuned models were titled CNNForWord2VecBERTFT, CNNForWord2VecBERTweetFT, and CNNForWord2VecALBERTFT.

Additional Components	Output Layer
CNN, Attention, Frozen BERT Layer	Concatenation, FC

Fig 4. Models fine tuned with attention and frozen BERT layer

An Optuna Study of 5 trials was performed for all nine models with the objective value being f1 score due to the equal importance of precision and recall. On the right is an analysis of the Study for the best performing model, CNNForWord2VecBertTweet.

The best hyperparameters for each study were used for model training.

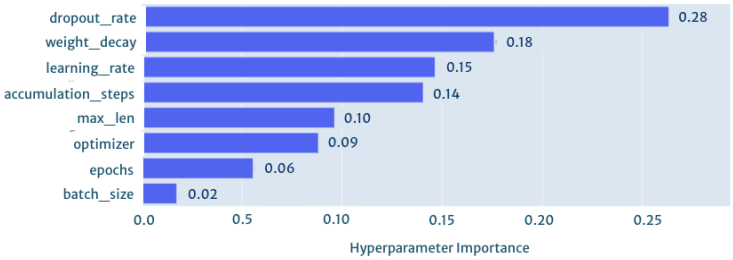


Fig 5. Hyperparameter importance after Optuna trials

4 Results

4.1 Comparison of Results

Metrics		train_accuracy	train_precision	train_recall	train_f1	val_accuracy	val_precision	val_recall	val_f1
W2v+BERT	Base	0.9536	0.952875	0.9544	0.953637	0.849167	0.908382	0.776667	0.837376
BerTweet only	baseline	0.9694	0.969964	0.9688	0.969382	0.87	0.912639	0.818333	0.862917
W2v+BERT	FT	0.9474	0.947579	0.9472	0.947389	0.8625	0.860697	0.865	0.862843
W2v+BERTweet	FT	0.9958	0.994022	0.9976	0.995808	0.853333	0.856902	0.848333	0.852596
W2v+AlBERT	Base	0.8942	0.889988	0.8996	0.894768	0.84	0.795652	0.915	0.851163
W2v+BERTweet	Base	0.9896	0.989992	0.9892	0.989596	0.835833	0.85918	0.803333	0.830319
BERT only	baseline	0.987	0.986028	0.988	0.987013	0.8325	0.813187	0.863333	0.83751
Albert only	baseline	0.6946	0.703301	0.6732	0.687922	0.771667	0.807547	0.713333	0.757522
W2v+AlBERT	FT	0.6776	0.679903	0.6712	0.675523	0.67	0.81677	0.438333	0.570499

Fig 6. Model results on train and validation set

Model Name	Precision	Accuracy	Recall	F1-Score
CNNForWord2VecBERTweet	0.951	0.89	0.823	0.882
CNNForWord2VecBERT	0.927	0.896	0.859	0.892
BERTweet	0.899	0.898	0.896	0.897
CNNForWord2VecALBERT	0.827	0.87	0.935	0.878

Fig 7. Results for best performing models on the test set

When choosing the best model for sarcasm detection, precision was the key focus to minimize false positives and avoid misunderstandings. The “CNNForWord2VecBERTweet” model scored higher in precision, as it used custom Word2Vec embeddings that capture subtle linguistic cues of sarcasm effectively combined with the context-aware embeddings modified specifically for English tweets from the BERTweet model.

The precision choice is crucial in contexts like social media, where incorrectly labeling a non-sarcastic comment as sarcastic can lead to misunderstandings. High precision ensures that when a tweet is labeled as sarcastic, it is very likely to be so, which is particularly important in applications such as moderating online interactions or analyzing customer feedback.

It appeared the fine-tuned models did not perform as well as anticipated compared to the other models. Applying the attention mask for non-padded tokens may have been unnecessary since the BERT models already incorporate attention in their architecture. Additionally, freezing the BERT layer to place further emphasis on the Word2Vec embeddings may not have been as effective as updating the weights and biases during backpropagation.

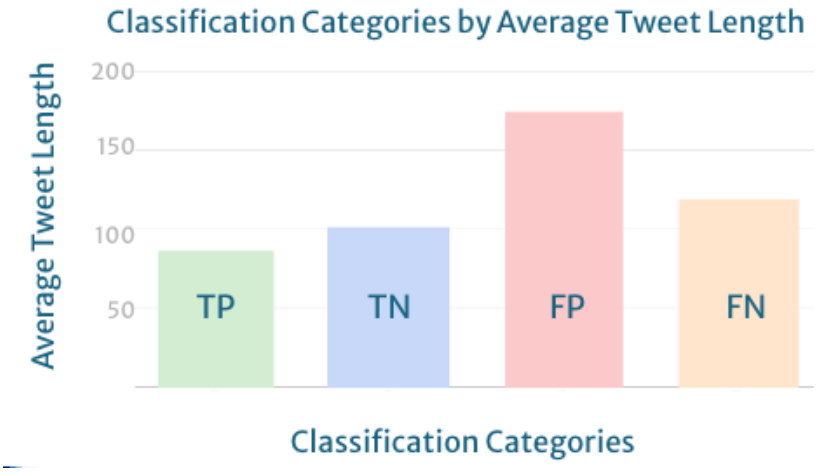


Fig 8. Classification Categories by Average Tweet Length

Further analysis of the actual tweets used in the test set showed that significantly longer tweets appeared to generate more false positive and false negative results from the model, indicating difficulties in the model to correctly classify longer tweets. With additional topic modeling analysis (Appendix E), it was found that most of the misclassifications were attributed to tweets about politics, violence, and religion.

4.2 Embeddings Analysis

A PCA analysis (Appendix C) found highly overlapping clusters, yielding little insight into the general distribution of similarity between the true and false predictions. In order to improve the visualization of the data, the following methodology was adopted:

1. Generate centroids of the embeddings for true positives and true negatives
2. Calculate cosine similarity scores for input embeddings with each class
3. Map the cosine similarity scores to columns: “similarity to opposite class” and “similarity to same class”
4. Visualize the embeddings’ cosine similarity scores, using their performance classification (i.e., TP, TN, FP, FN) as a basis for point coloration
5. Review the distribution of the points corresponding with FN and FP

The visualization in Figure 12 shows the following:

- **False positives** tend to cluster where “similarity to the same class” (SSC) < 0.25 and similarity to the opposite class (SSO) > 0.6.
- **False negatives** tend to cluster with false positives, but extend to a window where SSC < 0.2 and SSO > 0.6.

To maximize precision, we recommend the following adjustments:

1. Adjust the predicted labels, setting class 1 to class 0 where SSC < 0.25 and SSO > 0.6. This flip would improve precision substantially.
2. Adjust both labels where SSC < 0.25’ ;7 and SSO > 0.6

Below are the subsequent changes in performance on the test set given the adjustment to both classes. By following adjustment (2), recall improved **+0.09** to 0.91 and precision improved **+0.05**, achieving 1.0.

Accuracy: 0.89
Recall: 0.82
Precision: 0.95
F1 Score: 0.88
Classification Report:

	precision	recall	f1-score	support
0.0	0.84	0.96	0.90	1500
1.0	0.95	0.82	0.88	1500
accuracy			0.89	3000
macro avg	0.90	0.89	0.89	3000
weighted avg	0.90	0.89	0.89	3000

	Predicted Negative	Predicted Positive
Actual Negative	1436	64
Actual Positive	265	1235

Fig 9. Performance results before adjustment

Accuracy: 0.95
Recall: 0.91
Precision: 1.00
F1 Score: 0.95
Classification Report:

	precision	recall	f1-score	support
0.0	0.92	1.00	0.96	1500
1.0	1.00	0.91	0.95	1500
accuracy			0.95	3000
macro avg	0.96	0.96	0.95	3000
weighted avg	0.96	0.95	0.95	3000

	Predicted Negative	Predicted Positive
Actual Negative	1498	2
Actual Positive	133	1367

Fig 10. Performance results after adjustment

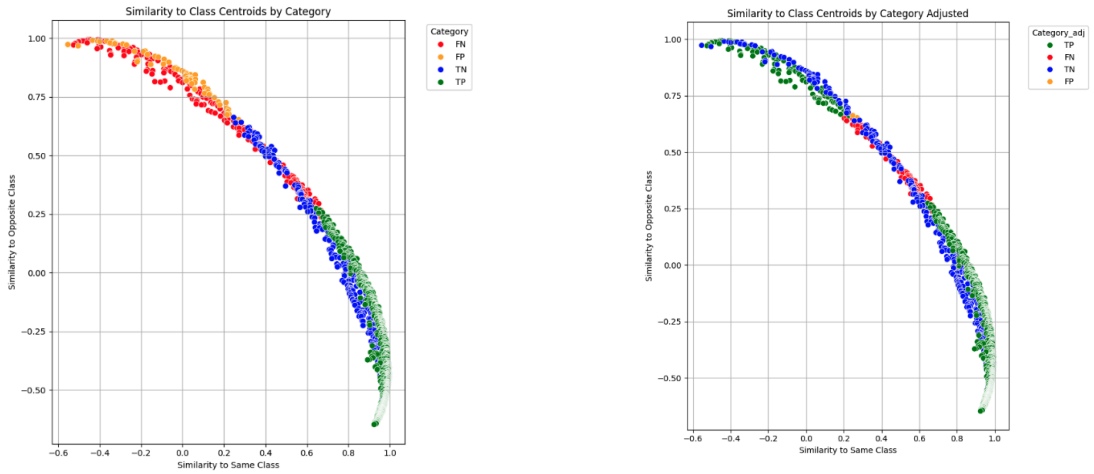


Fig 11. Embedding Analysis plots

5 Conclusion

Failure to detect sarcasm in tweets constitutes a fundamental issue for: (i) neurodivergent individuals coping with the challenges of communicating online, (ii) challenges to social media trust and safety teams enforcing content policies and (iii) business applications relying on accurate representations of sentiment. This paper explored the efficacy of using large and small language models in combination with the concatenation of BERT and Word2Vec embeddings to classify tweets for sarcasm.

The performance of our top performing models exceeded the performance of the models cited in UTNLP at SemEval-2022 Task 6: A Comparative Analysis of Sarcasm Detection using generative-based and mutation-based data augmentation; however, we acknowledged that our data was substantially smaller and limited in scope (i..e., tweets). The highest performing model on the test set was the CNNForWord2VecBERTweet, whose performance was substantially improved by a predicted label adjustment using the output embedding similarities to the same and opposite class centroids (SSC and SSO, respectively).

Below are the recommendations for additional work in this space:

- Perform topic modeling (using BERTopic or Latent Dirichlet Allocation) on false negative and false positive cases.
- Using topic modeling results and manual review to identify opportunities for data augmentation where inputs are either under represented (i.e., sparse in number) or difficult to disambiguate (i.e., close to topics shared by both classes).
- Our dataset did not include emojis in the tweets. Emojis can be useful in identifying context within tweets to better classify sarcasm and further research can be done on the role of emojis in communicating sarcasm.

Due to the nature of sarcasm, it is foreseeable that determinant factors for sarcasm will change as it is somewhat dependent on the public sentiment and personal circumstance. For this reason, it would be valuable to have a longitudinal study for understanding the classification of sarcasm over time and by/between generations.

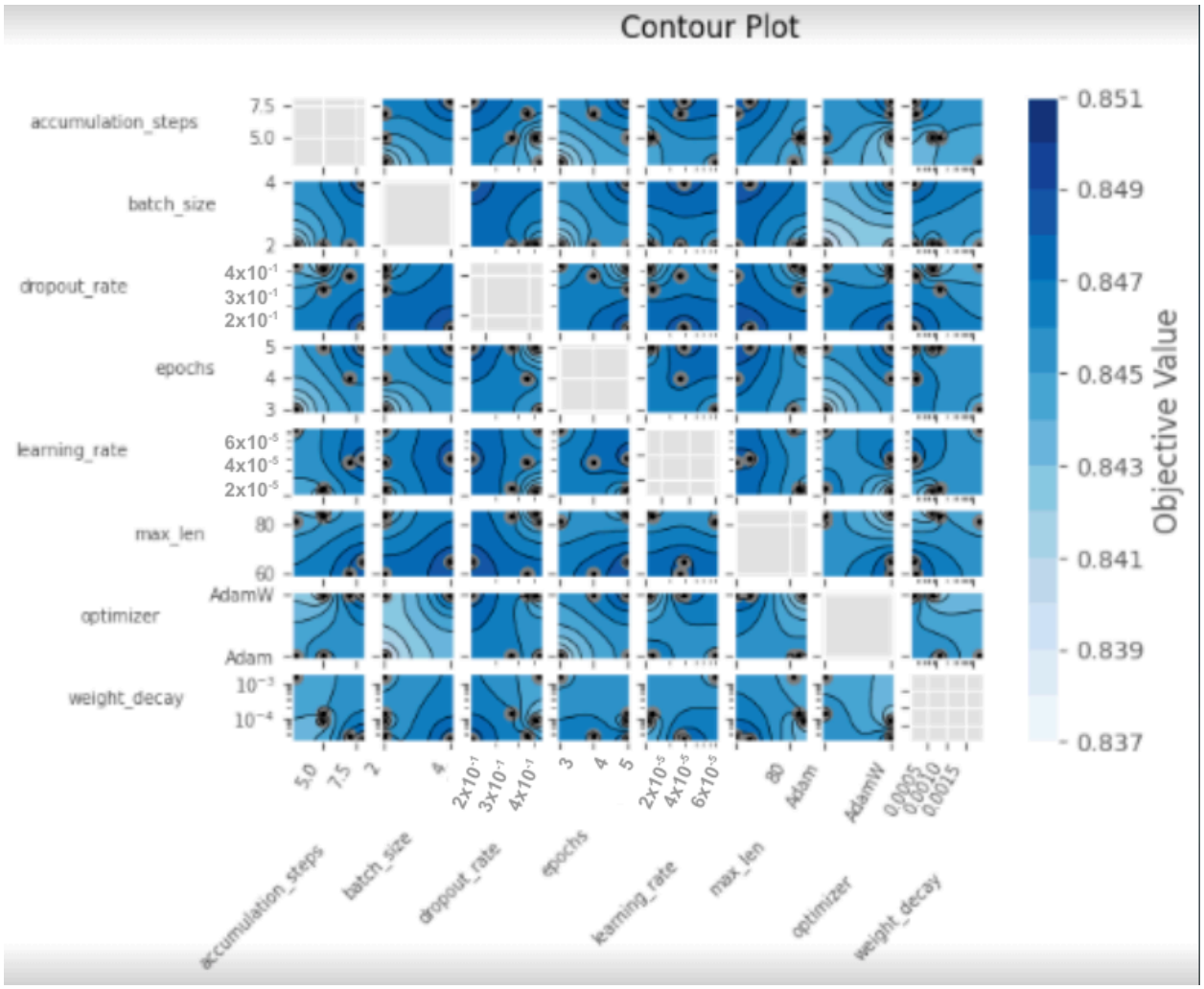
References

1. A. Abaskohi, A. Rasouli , T. Zeraati , B. Bahrak, *UTNLP at SemEval-2022 Task 6: A Comparative Analysis of Sarcasm Detection Using Generative-based and Mutation-based Data Augmentation*; School of Electrical and Computer Engineering, College of Engineering, University of Tehran
2. T. Sosea, J. Jessy Li, C. Caragea, *Sarcasm Detection in a Disaster Context*; Department of Computer Science, University of Illinois Chicago; Department of Linguistics, The University of Texas at Austin
3. D. Quoc Nguyen , T. Vu, A. Tuan Nguyen, *BERTweet: A pre-trained language model for English Tweets*; VinAI Research, Vietnam; Oracle Digital Assistant, Oracle, Australia; NVIDIA, USA
4. I. Alghanmi, L. Espinosa-Anke, S. Schockaert, *Combining BERT with Static Word Embeddings for Categorizing Social Media*; Cardiff University, UK
5. Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, *ALBERT: A Lite BERT For Self-Supervised Learning Of Language Representations*; Google Research, Toyota Technological Institute at Chicago
6. J. Devlin, M. Chang, K. Lee, K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*; Google AI Language
7. T. Mikolov, K. Chen, G. Corrado, J. Dean, *Efficient Estimation of Word Representations in Vector Space*; Google Inc., Mountain View, CA

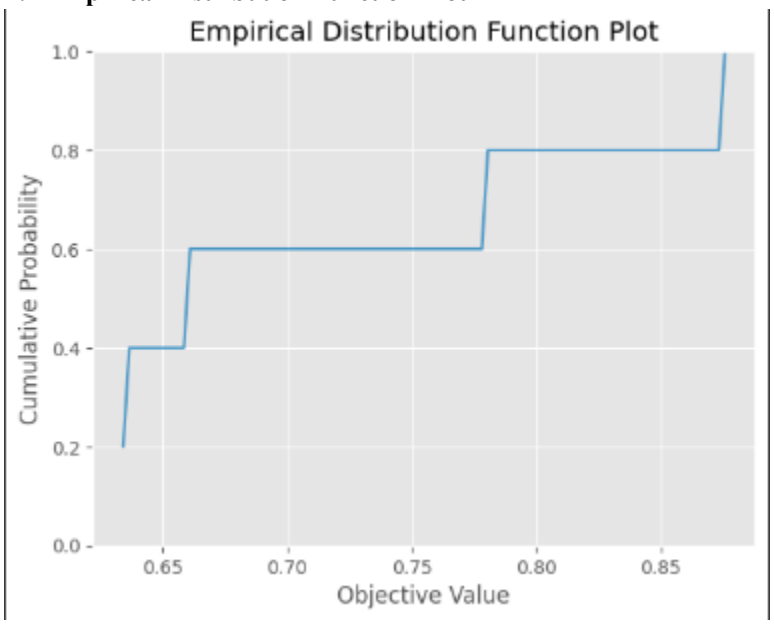
Appendix

A Hyperparameter Optimization Analysis for CNNForWord2VecBERTweet

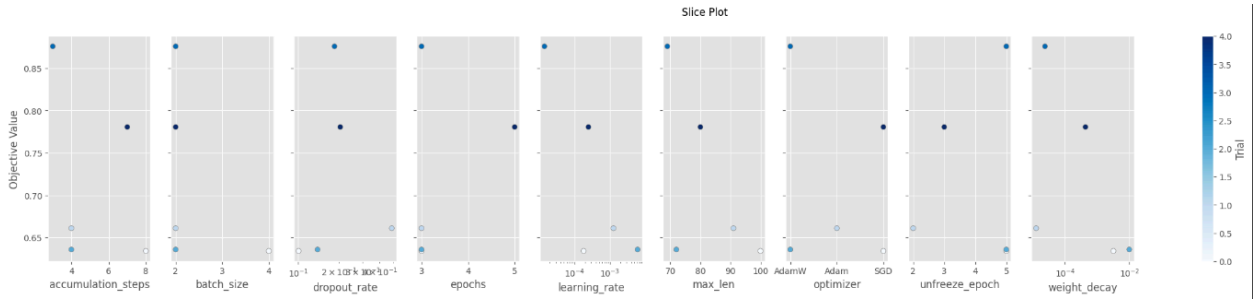
A.1 Contour Plot



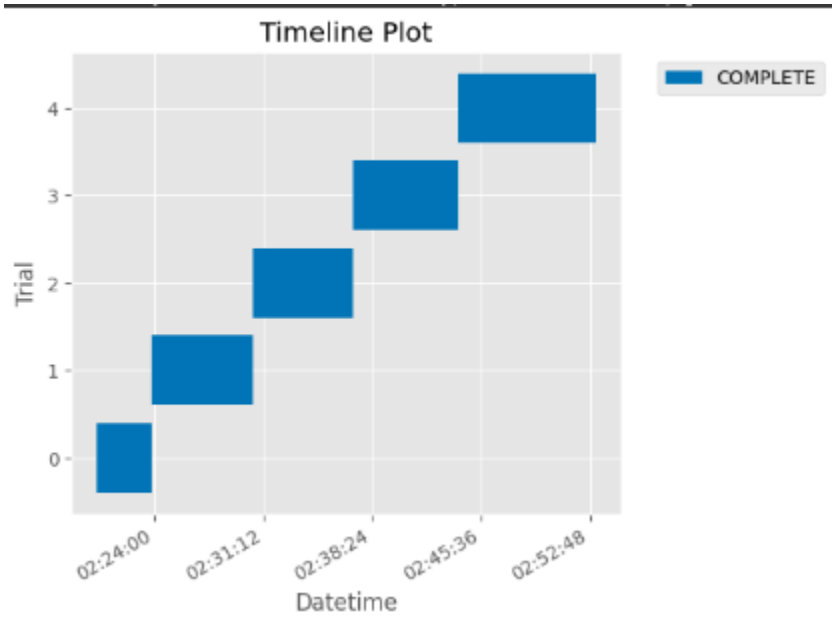
A.2 Empirical Distribution Function Plot



A.3 Slice Plot

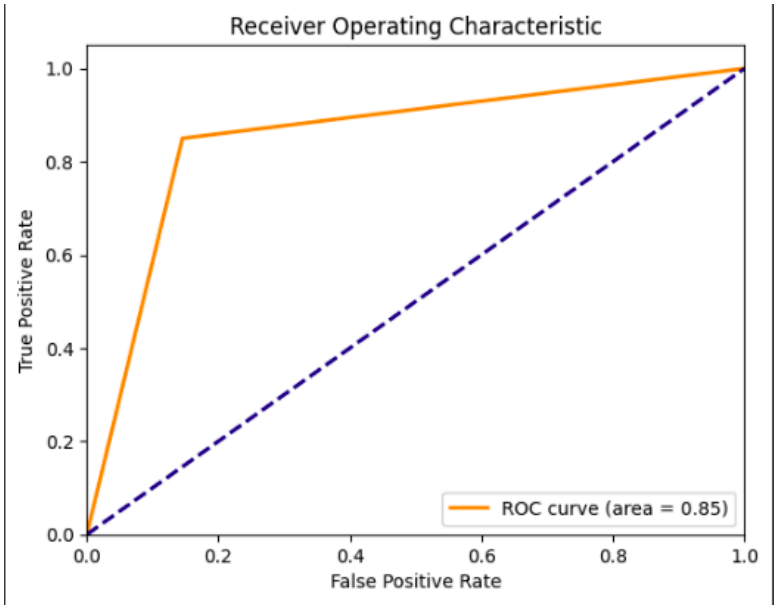


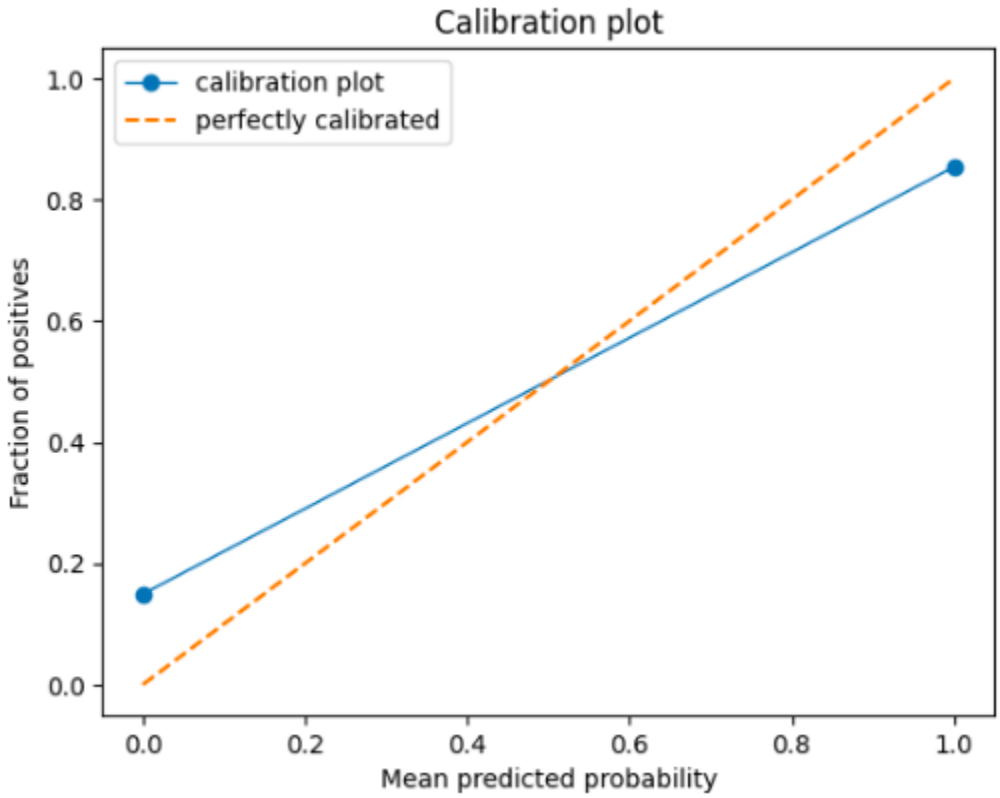
A.4 Timeline Plot

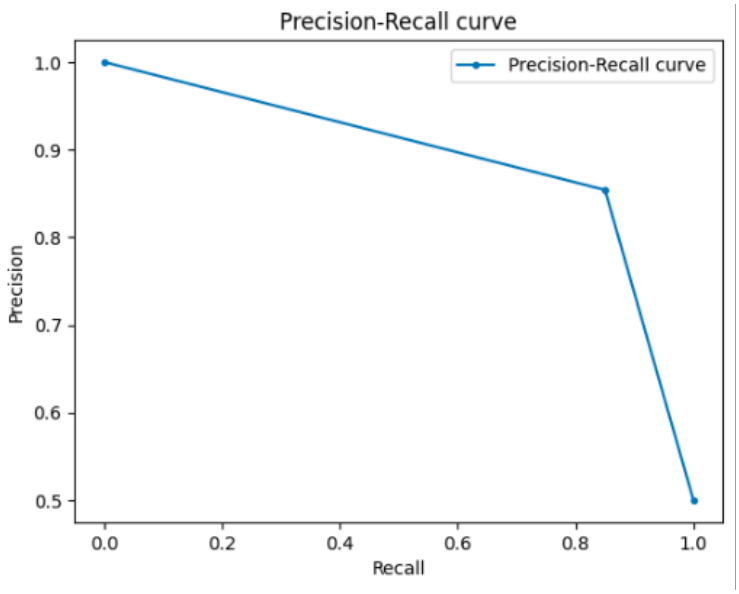


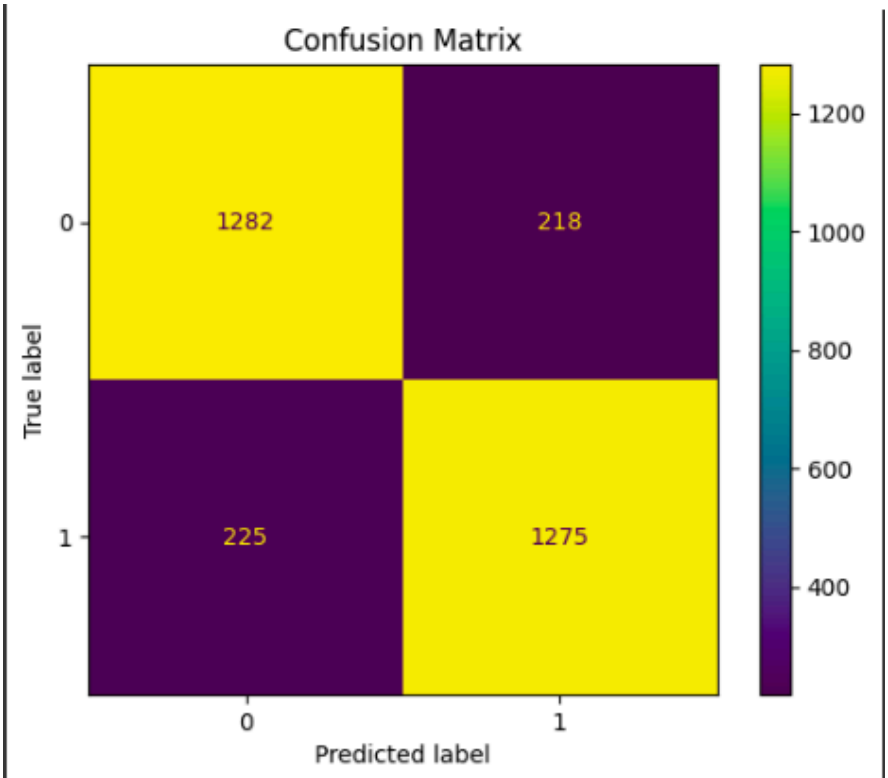
B Prediction Results for Each Model

B.1 BERT



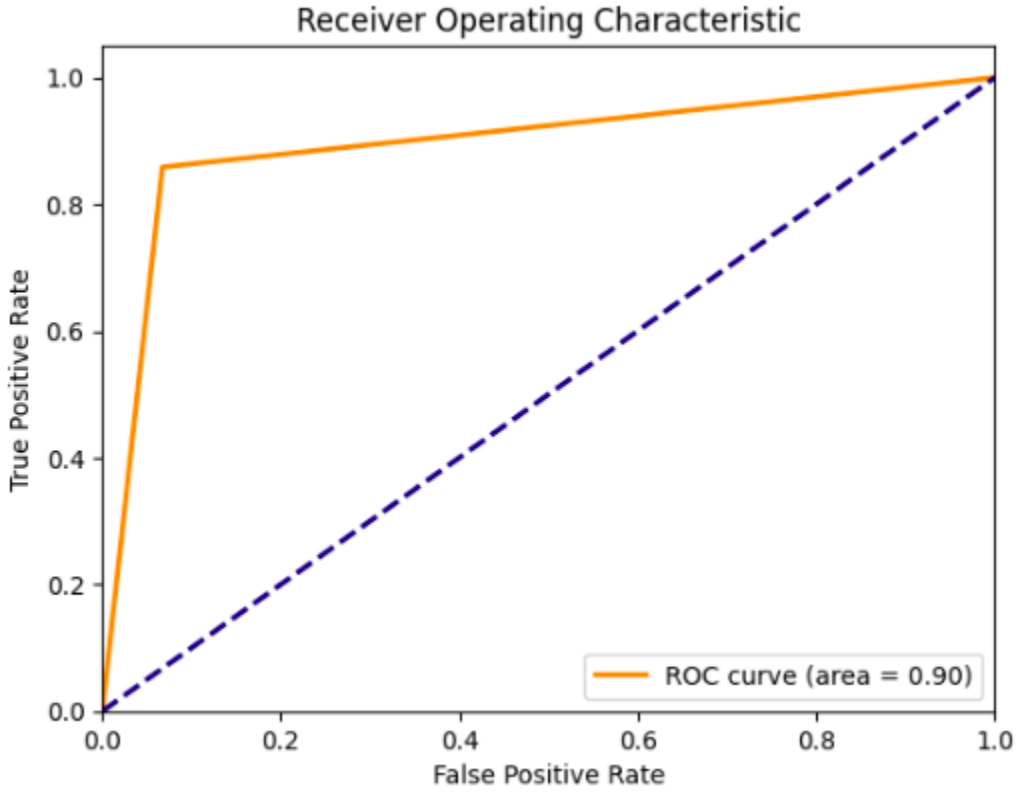


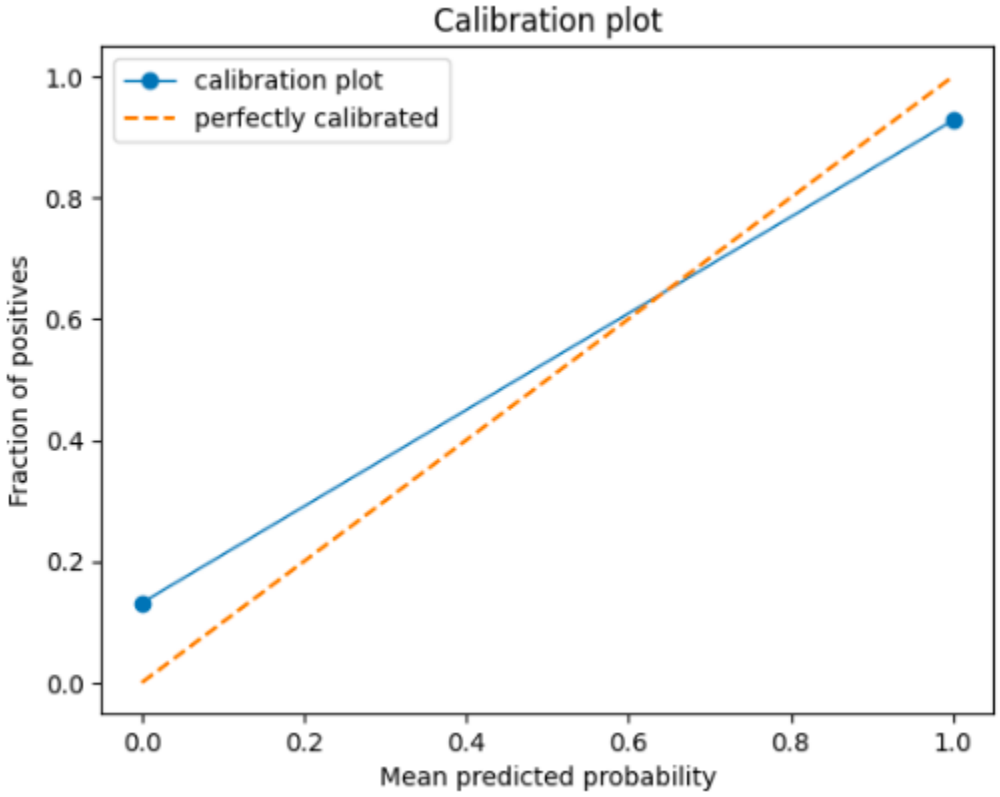


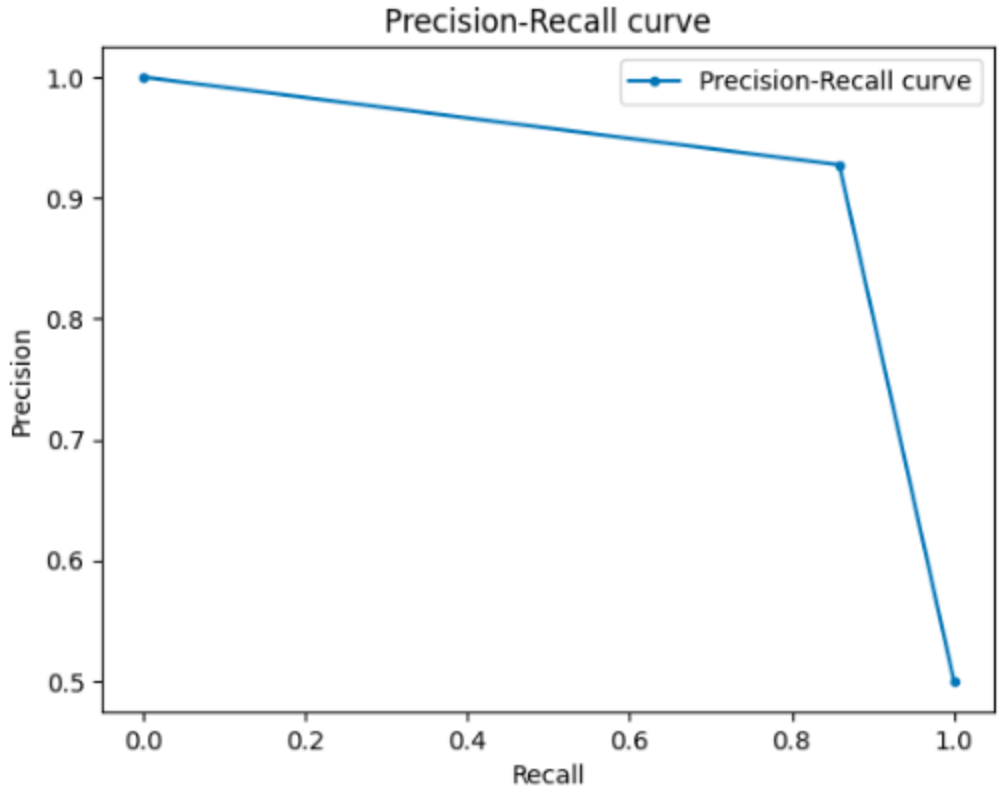


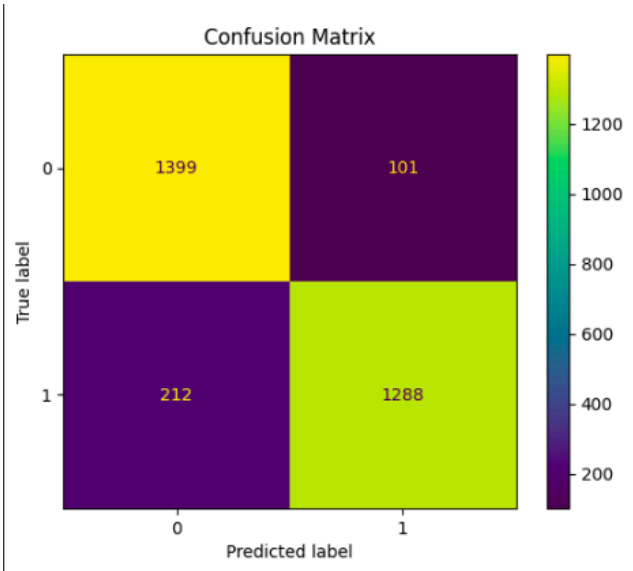
	precision	recall	f1-score	support
0.0	0.85	0.85	0.85	1500
1.0	0.85	0.85	0.85	1500
accuracy			0.85	3000
macro avg	0.85	0.85	0.85	3000
weighted avg	0.85	0.85	0.85	3000

B.2 CNNForWord2VecBERT



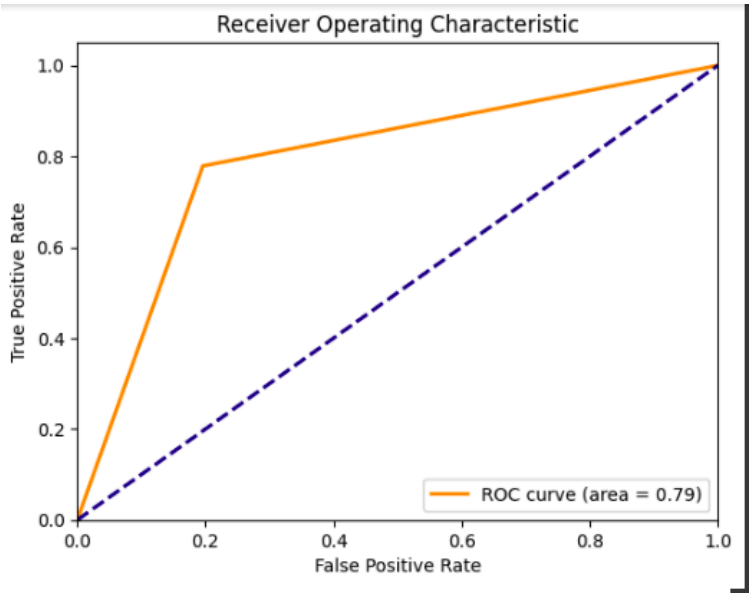


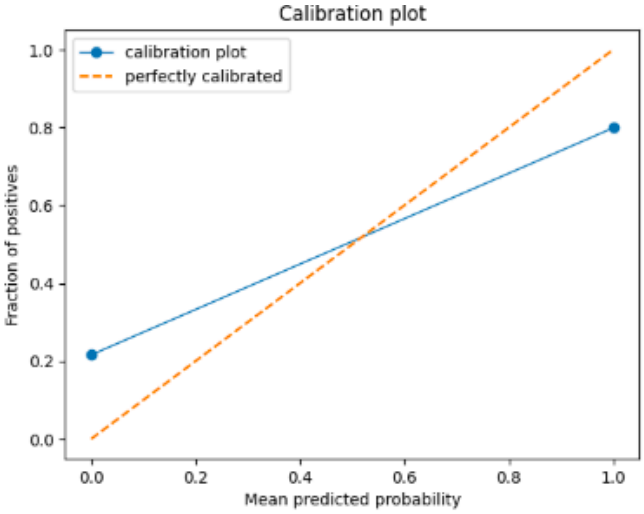


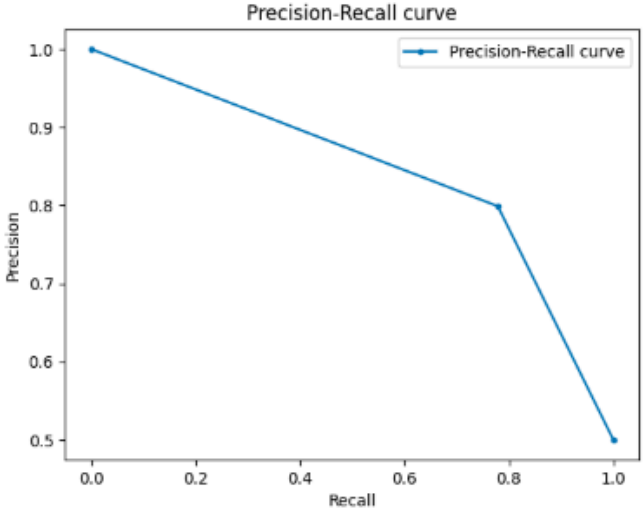


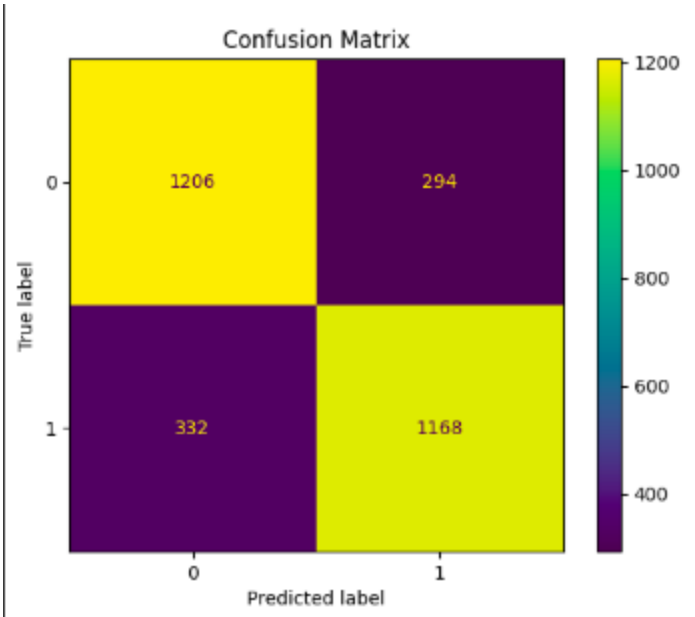
	precision	recall	f1-score	support
0.0	0.87	0.93	0.90	1500
1.0	0.93	0.86	0.89	1500
accuracy			0.90	3000
macro avg	0.90	0.90	0.90	3000
weighted avg	0.90	0.90	0.90	3000

B.3 CNNForWord2VecBERTFT



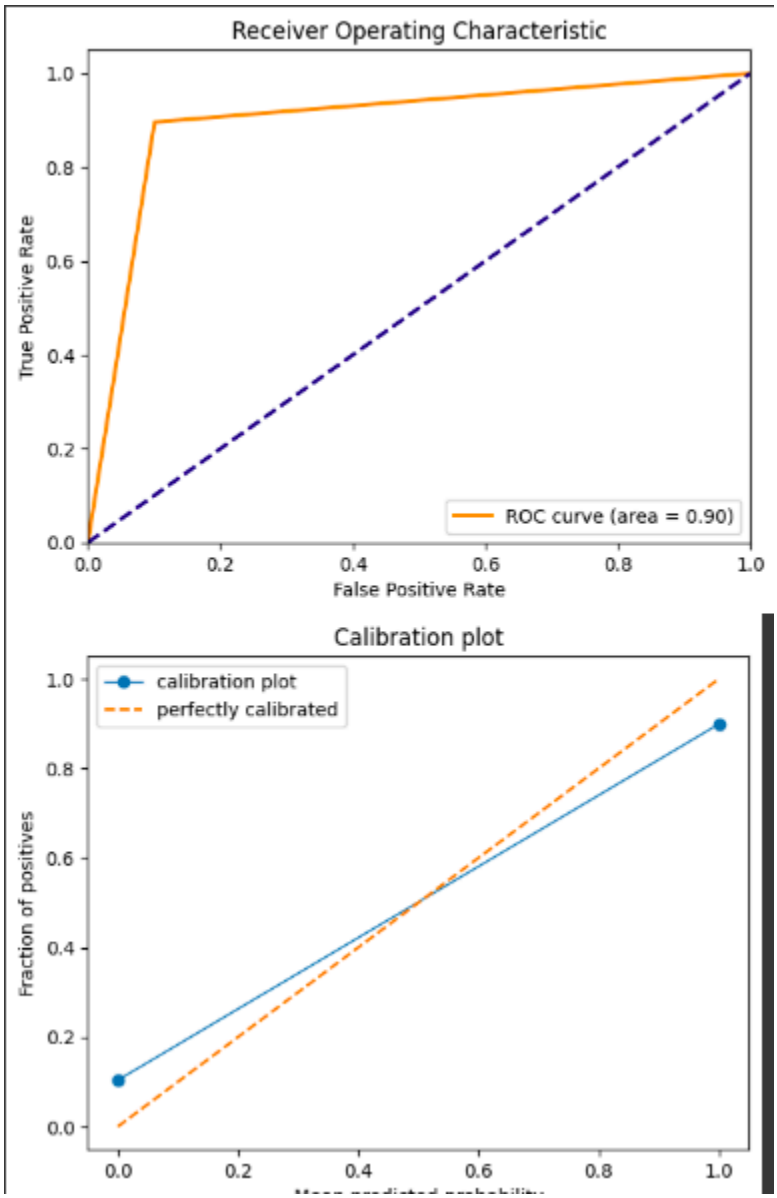


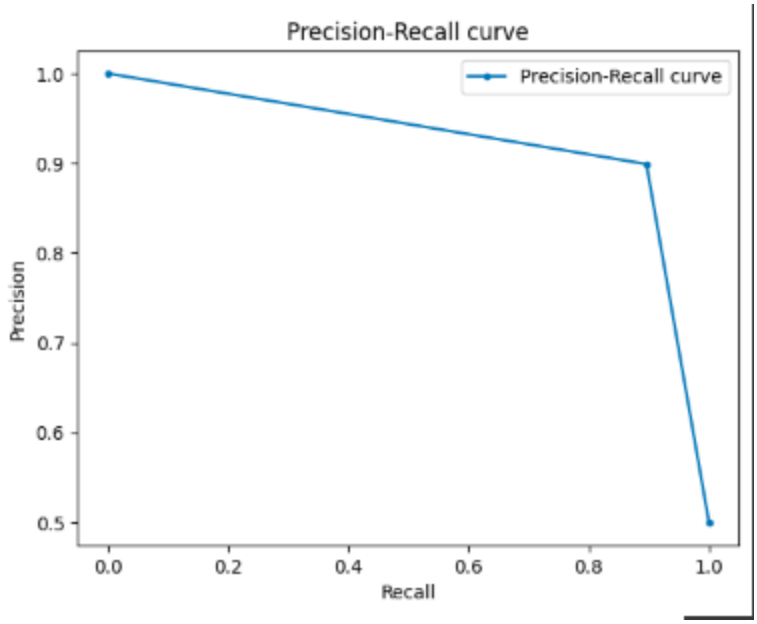


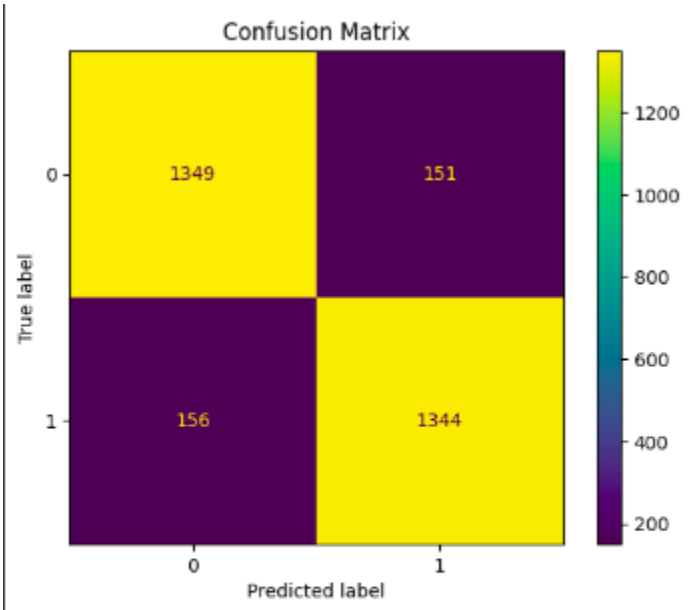


	precision	recall	f1-score	support
0.0	0.78	0.80	0.79	1500
1.0	0.80	0.78	0.79	1500
accuracy			0.79	3000
macro avg	0.79	0.79	0.79	3000
weighted avg	0.79	0.79	0.79	3000

B.4 BERTweet

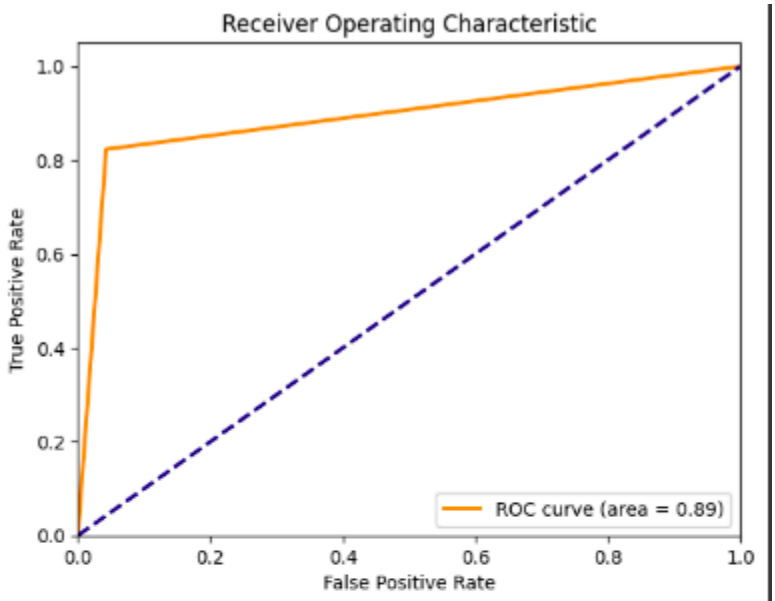

Receiver Operating Characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.90)
0.0
0.2
0.4
0.6
0.8
1.0
Calibration plot
calibration plot
perfectly calibrated
Fraction of positives

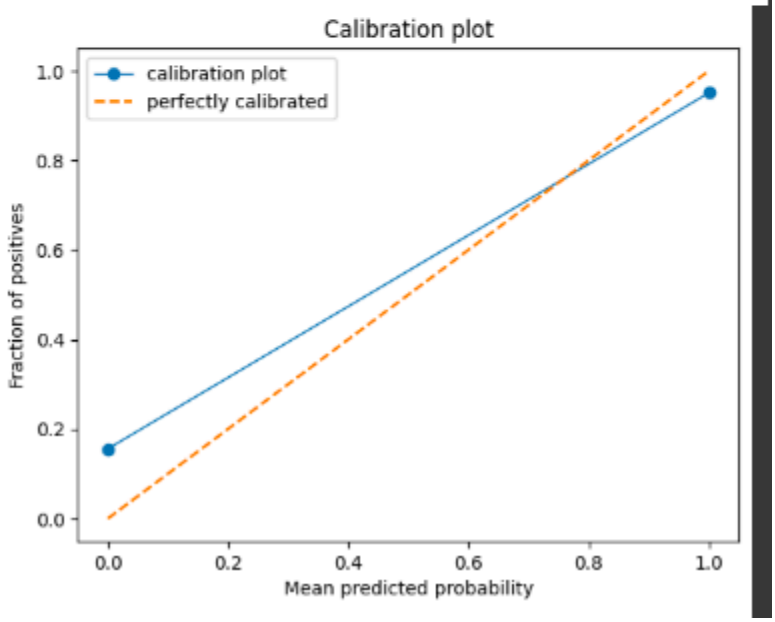


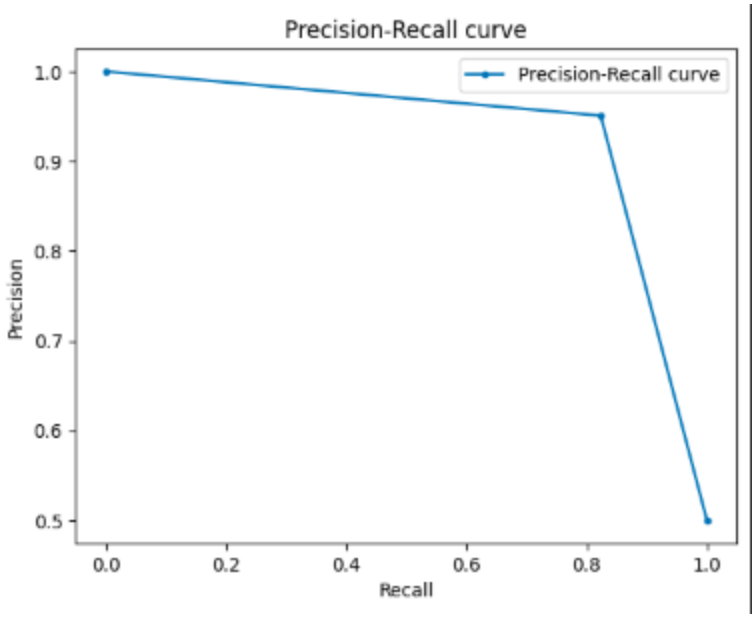


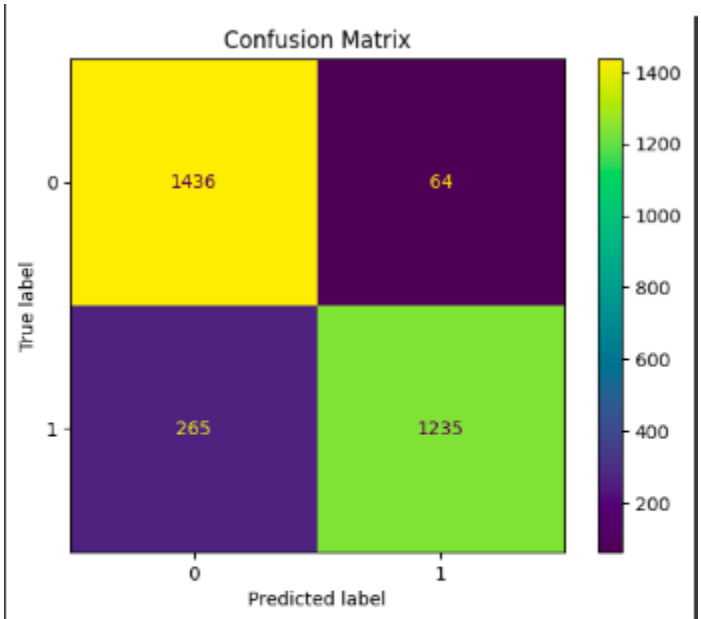
	precision	recall	f1-score	support
0.0	0.90	0.90	0.90	1500
1.0	0.90	0.90	0.90	1500
accuracy			0.90	3000
macro avg	0.90	0.90	0.90	3000
weighted avg	0.90	0.90	0.90	3000

B.5 CNNForWord2VecBERTweet



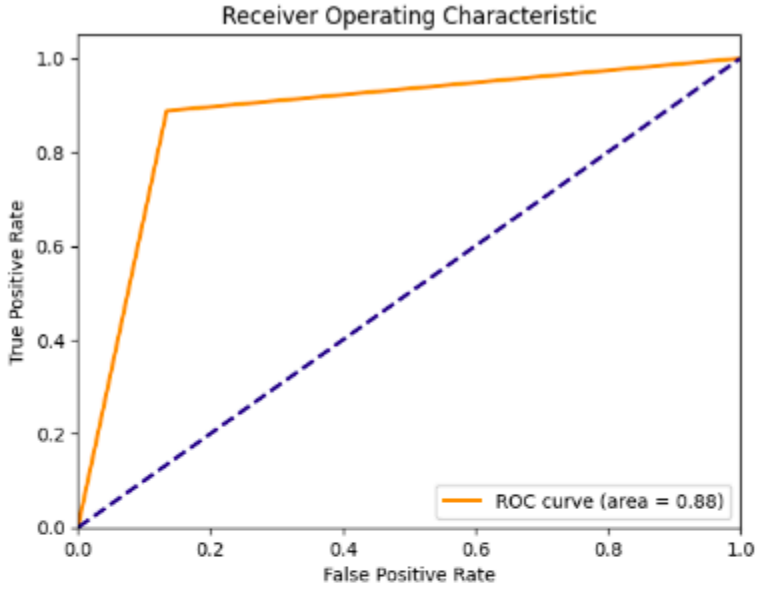


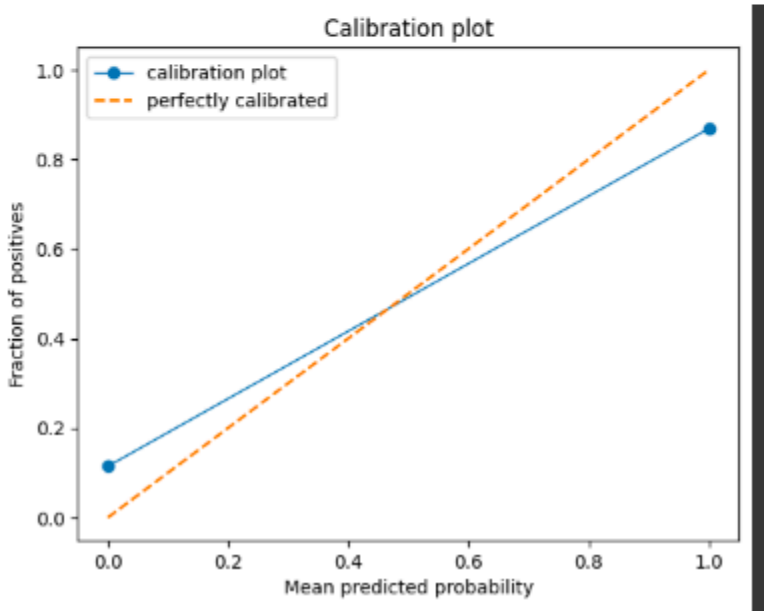


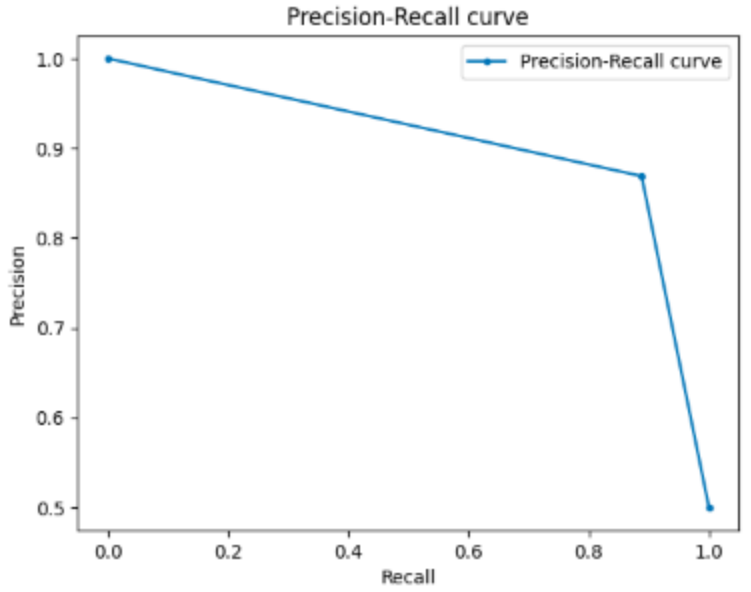


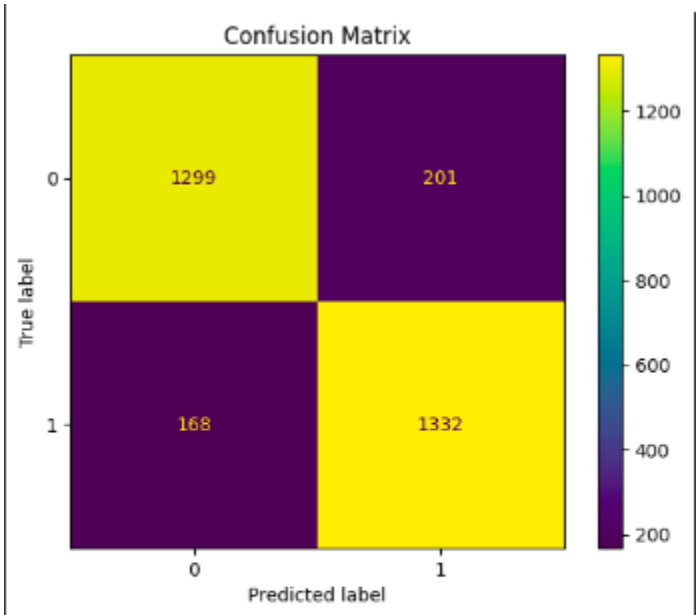
	precision	recall	f1-score	support
0.0	0.84	0.96	0.90	1500
1.0	0.95	0.82	0.88	1500
accuracy			0.89	3000
macro avg	0.90	0.89	0.89	3000
weighted avg	0.90	0.89	0.89	3000

B.6 CNNForWord2VecBERTweetFT



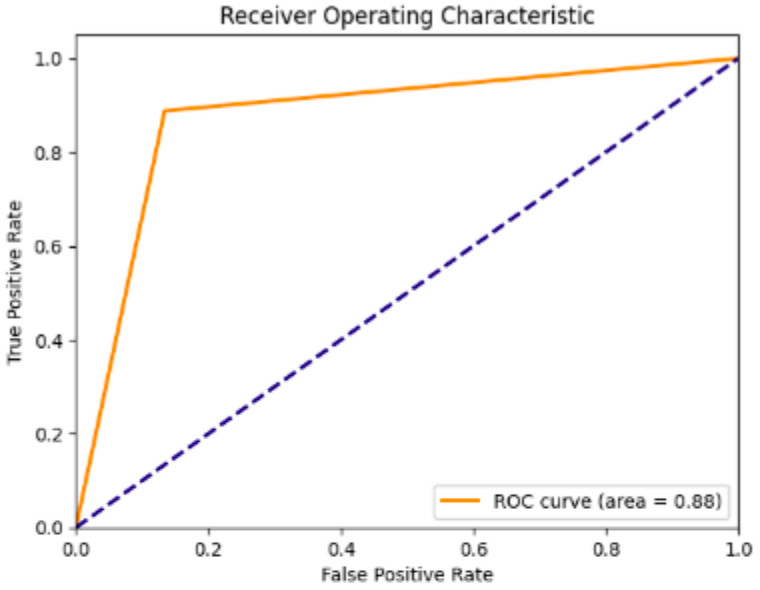


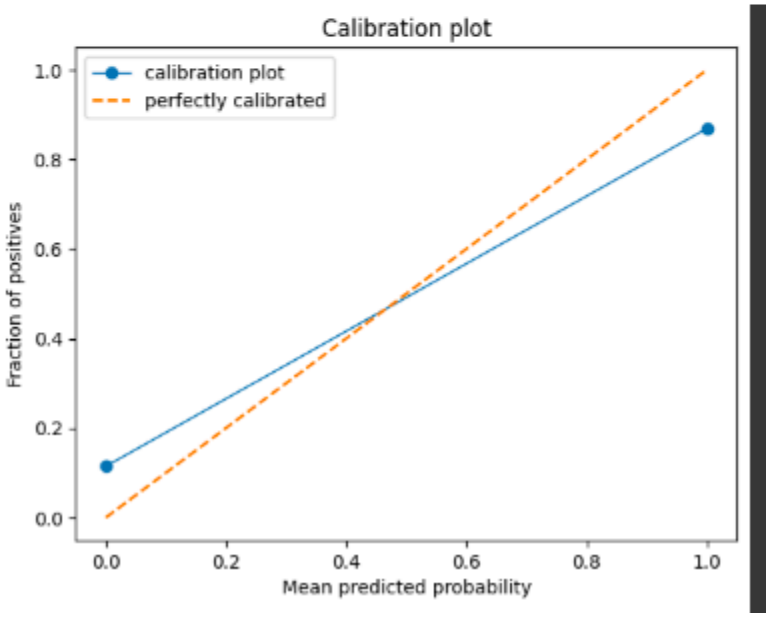


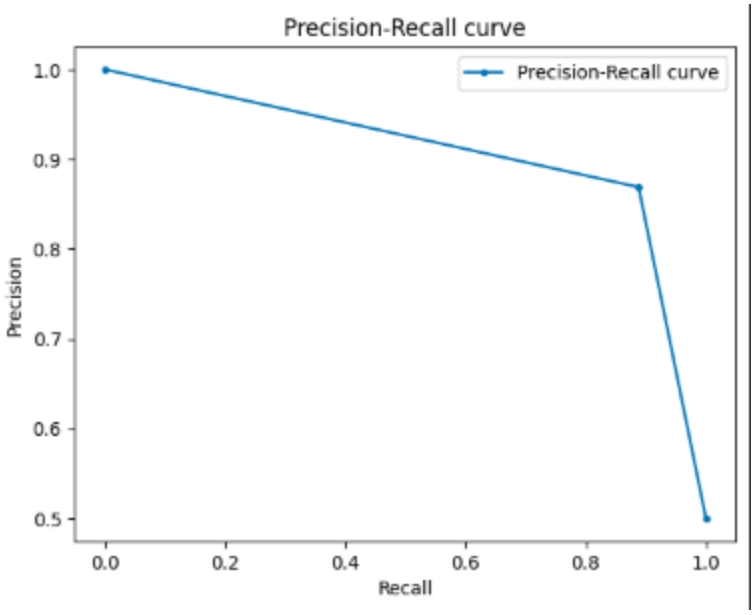


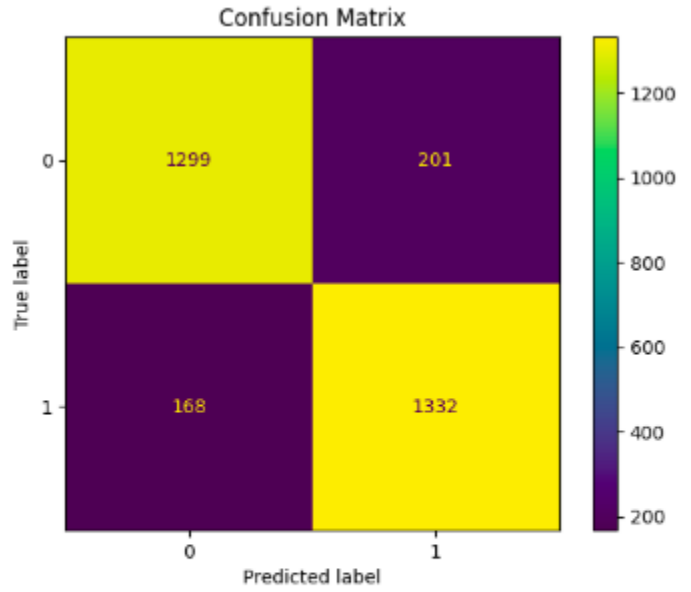
	precision	recall	f1-score	support
0.0	0.89	0.87	0.88	1500
1.0	0.87	0.89	0.88	1500
accuracy			0.88	3000
macro avg	0.88	0.88	0.88	3000
weighted avg	0.88	0.88	0.88	3000

B.7 ALBERT


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.88)
0.0
0.2
0.4
0.6
0.8
1.0

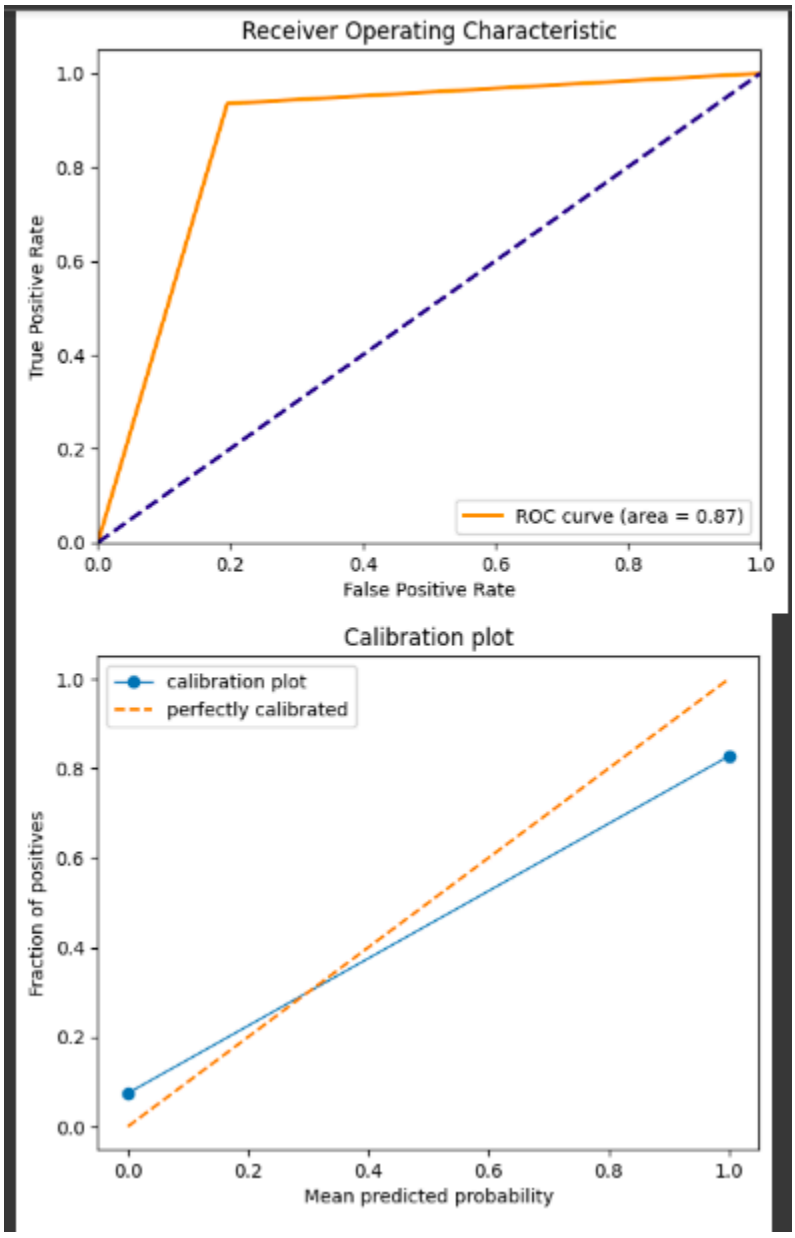

Calibration plot
calibration plot
perfectly calibrated
Fraction of positives
Mean predicted probability
0.0
0.2
0.4
0.6
0.8
1.0

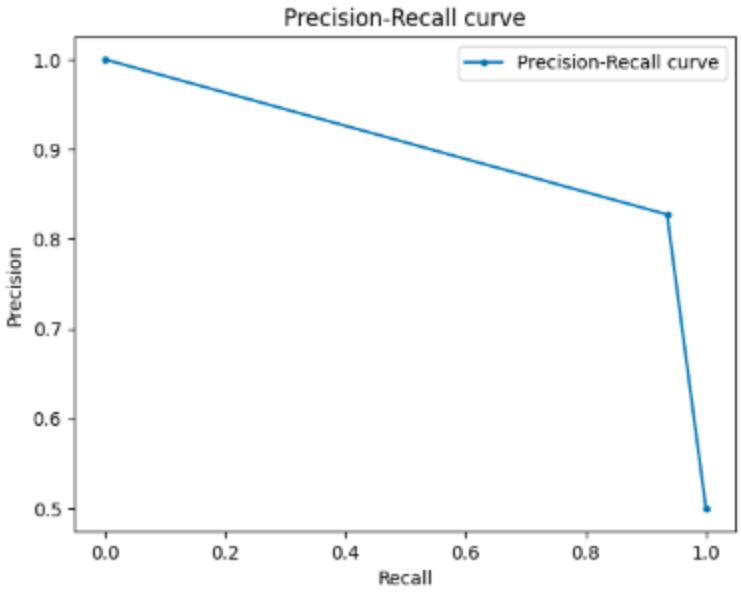


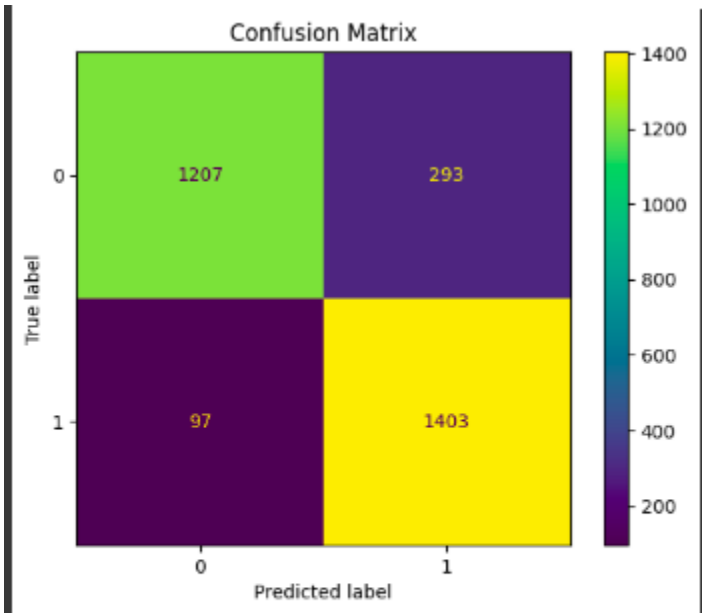


	precision	recall	f1-score	support
0.0	0.89	0.87	0.88	1500
1.0	0.87	0.89	0.88	1500
accuracy			0.88	3000
macro avg	0.88	0.88	0.88	3000
weighted avg	0.88	0.88	0.88	3000

B.8 CNNForWord2VecALBERT

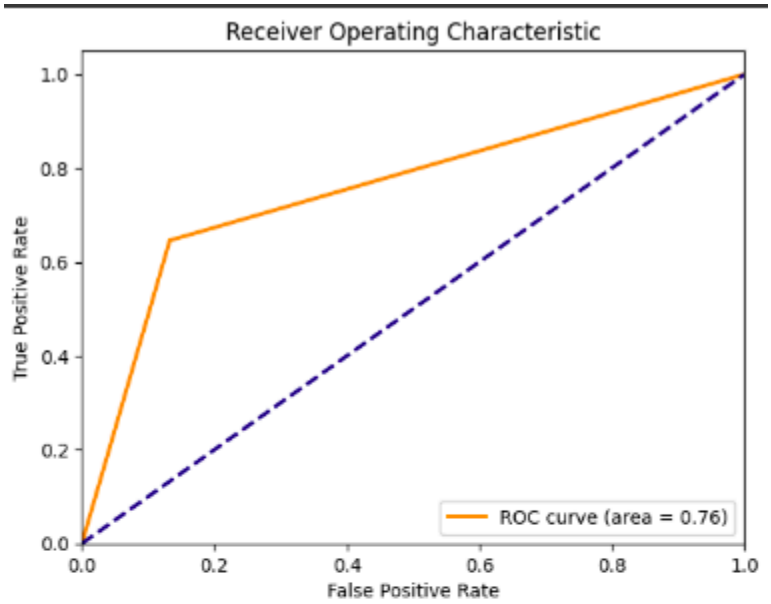


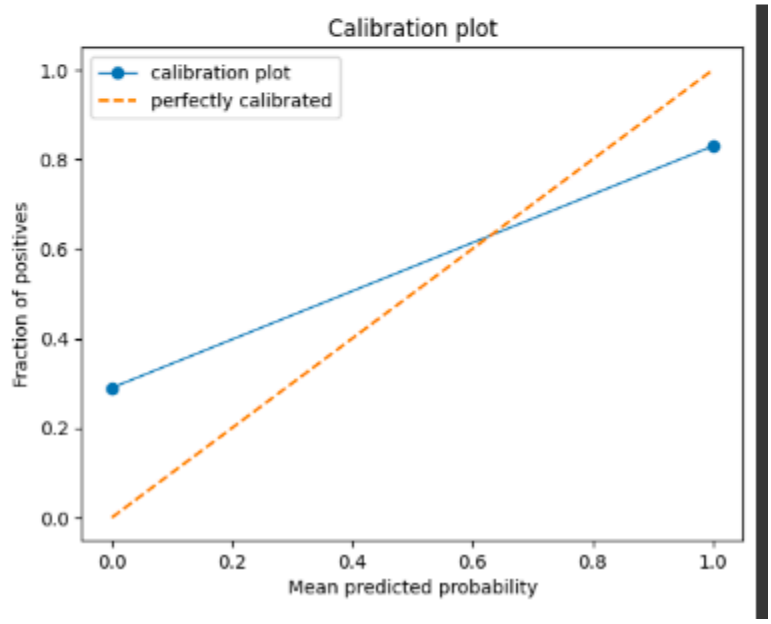


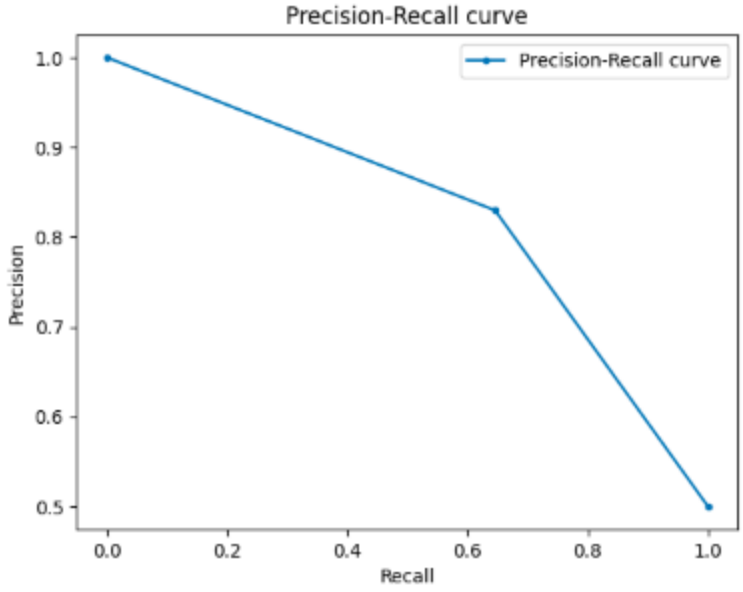


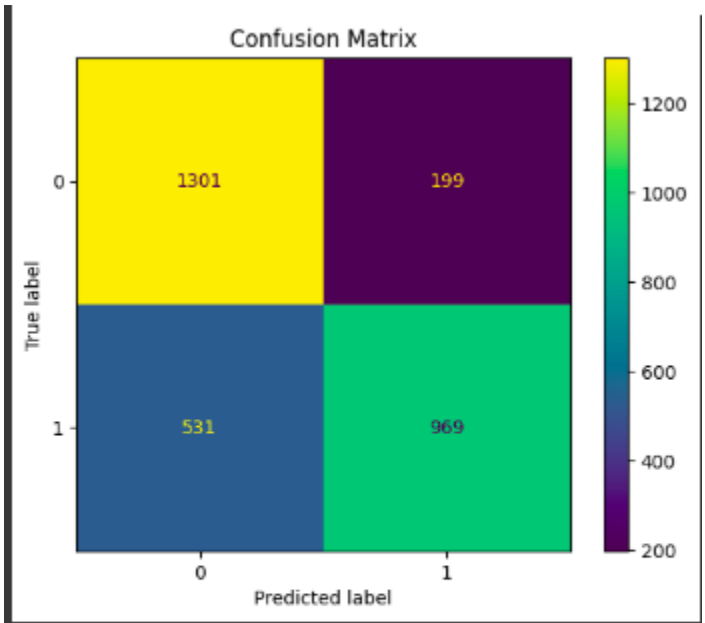
	precision	recall	f1-score	support
0.0	0.93	0.80	0.86	1500
1.0	0.83	0.94	0.88	1500
accuracy			0.87	3000
macro avg	0.88	0.87	0.87	3000
weighted avg	0.88	0.87	0.87	3000

B.9 CNNForWord2VecALBERTFT





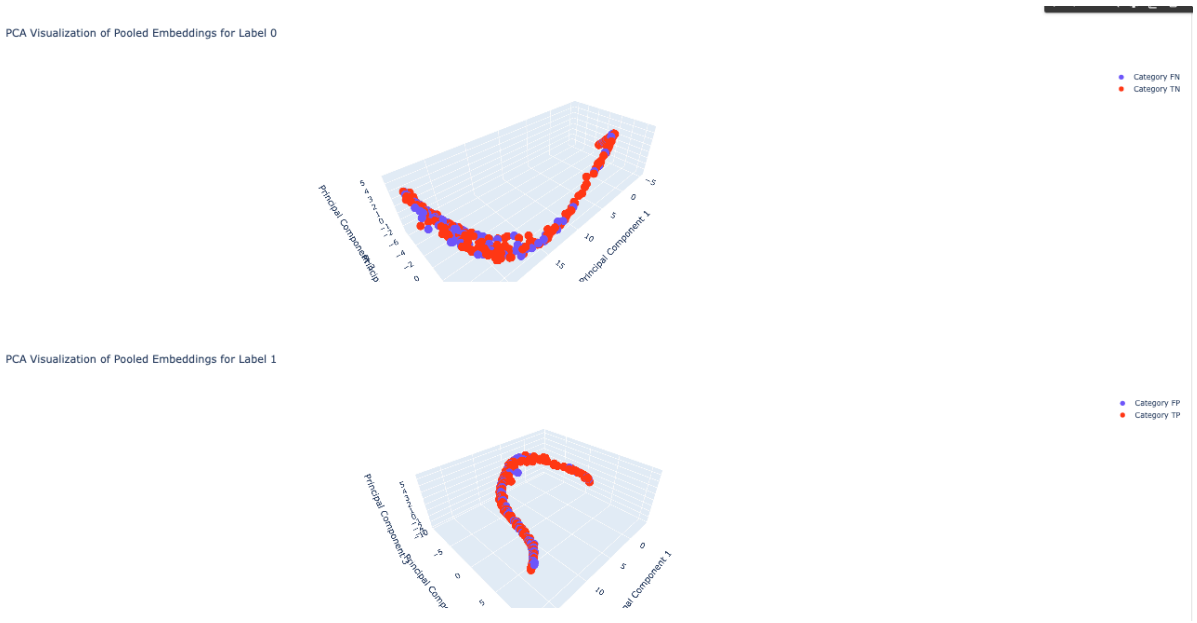




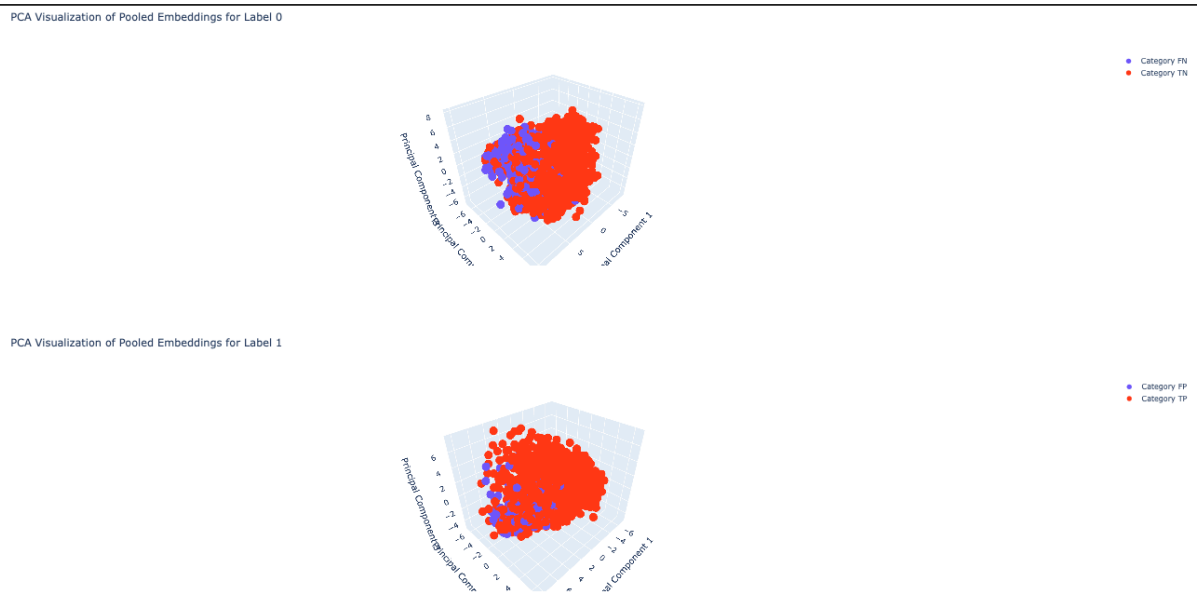
	precision	recall	f1-score	support
0.0	0.71	0.87	0.78	1500
1.0	0.83	0.65	0.73	1500
accuracy			0.76	3000
macro avg	0.77	0.76	0.75	3000
weighted avg	0.77	0.76	0.75	3000

C Embedding Analysis for Each Model Using Principal Component Analysis

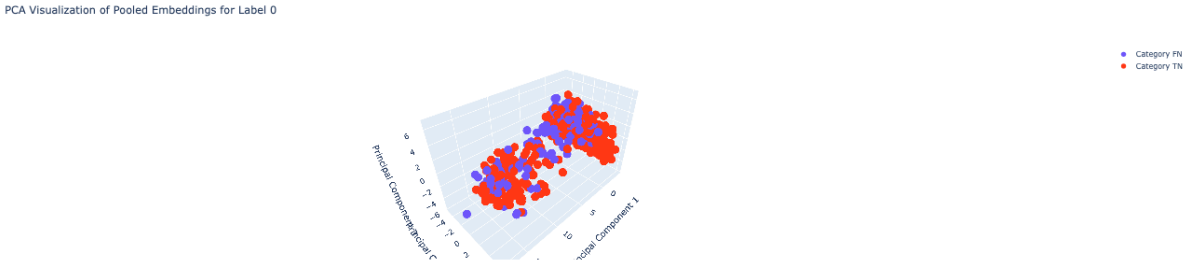
C.1 BERT

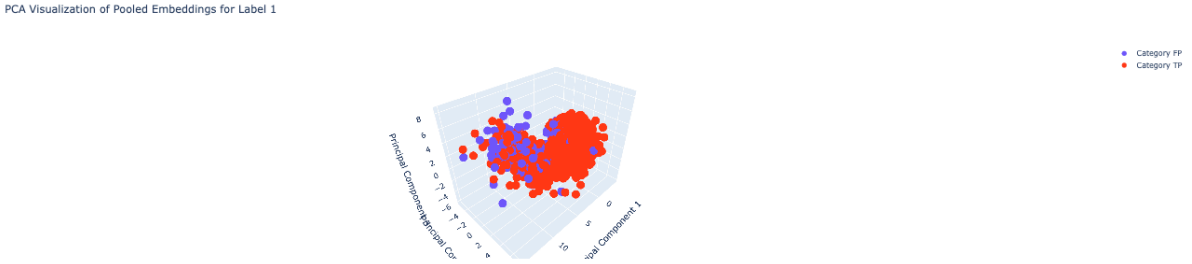


C.2 CNNForWord2VecBERT

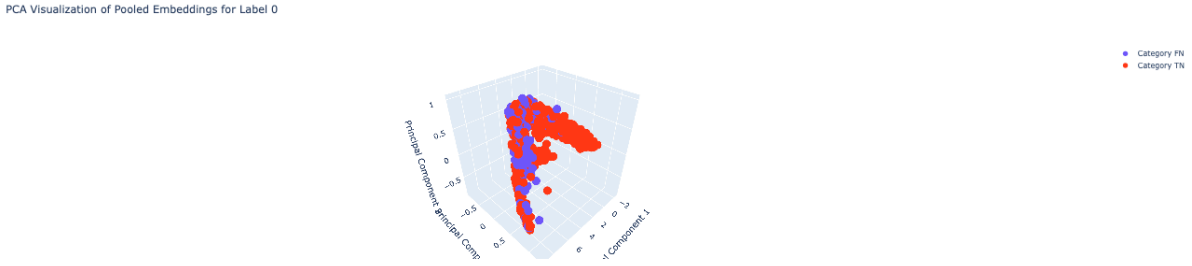


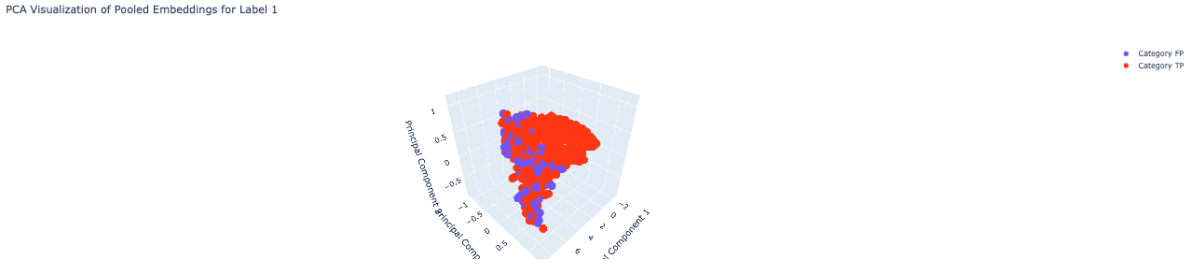
C.3 CNNForWord2VecBERTFT





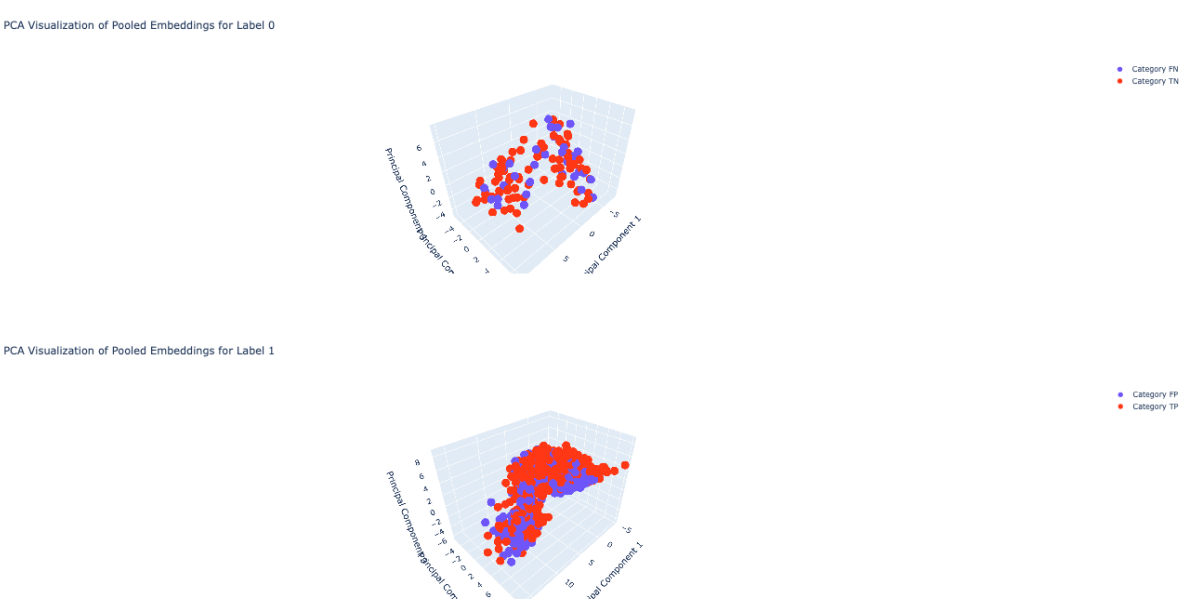
C.4 BERTweet





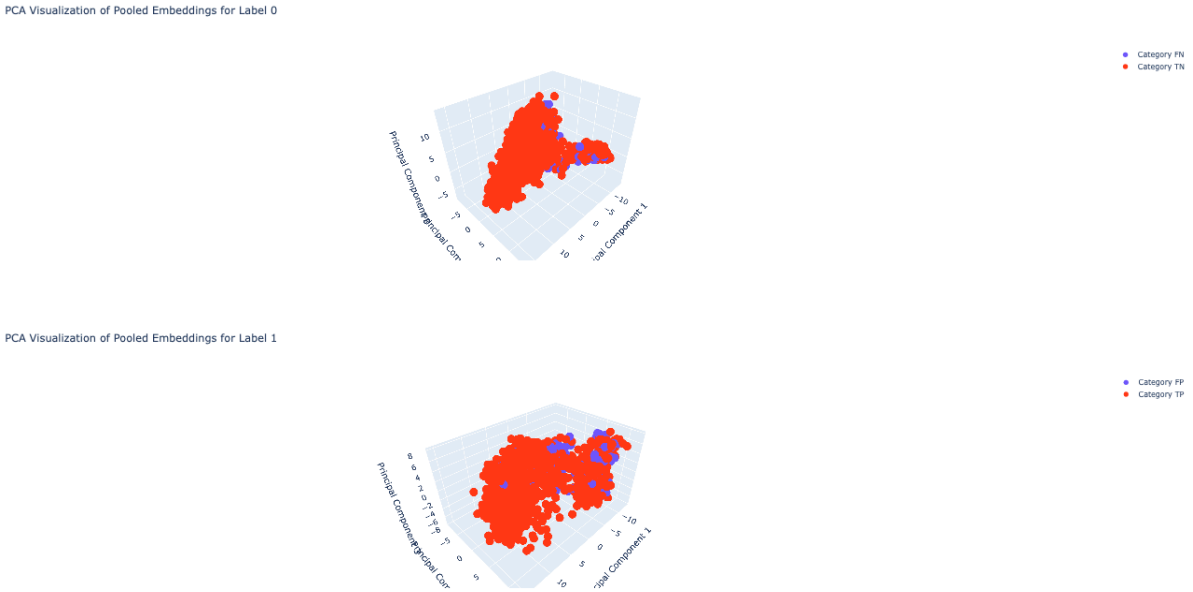
C.5 CNNForWord2VecBERTweet

C.6 CNNForWord2VecBERTweetFT

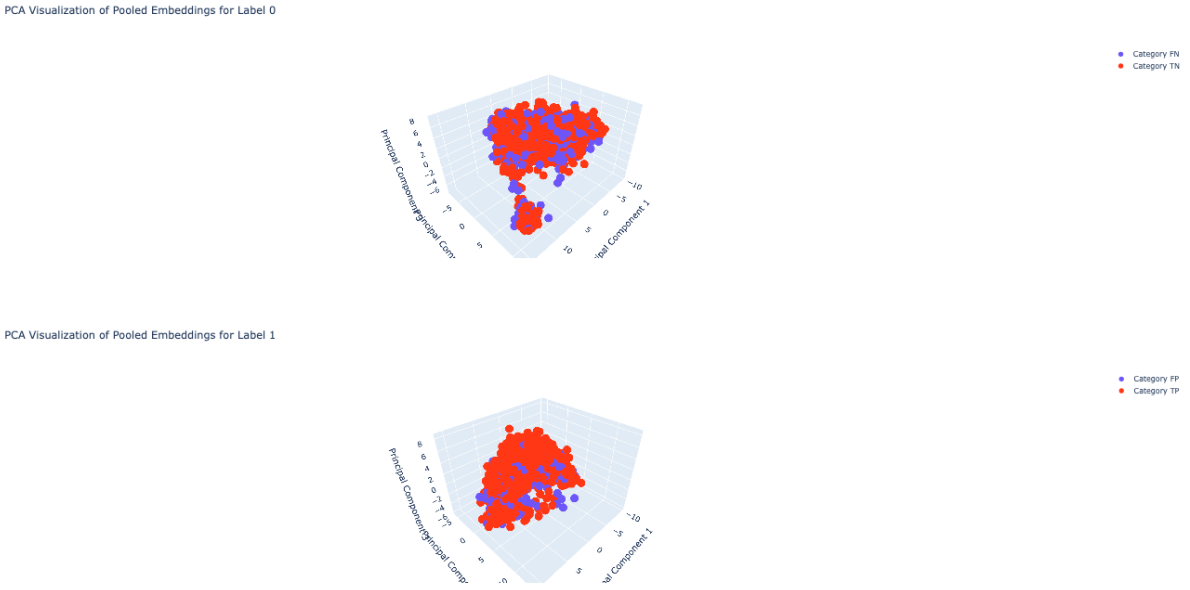


C.7 ALBERT

C.8 CNNForWord2VecALBERT



C.9 CNNForWord2VecALBERTFT

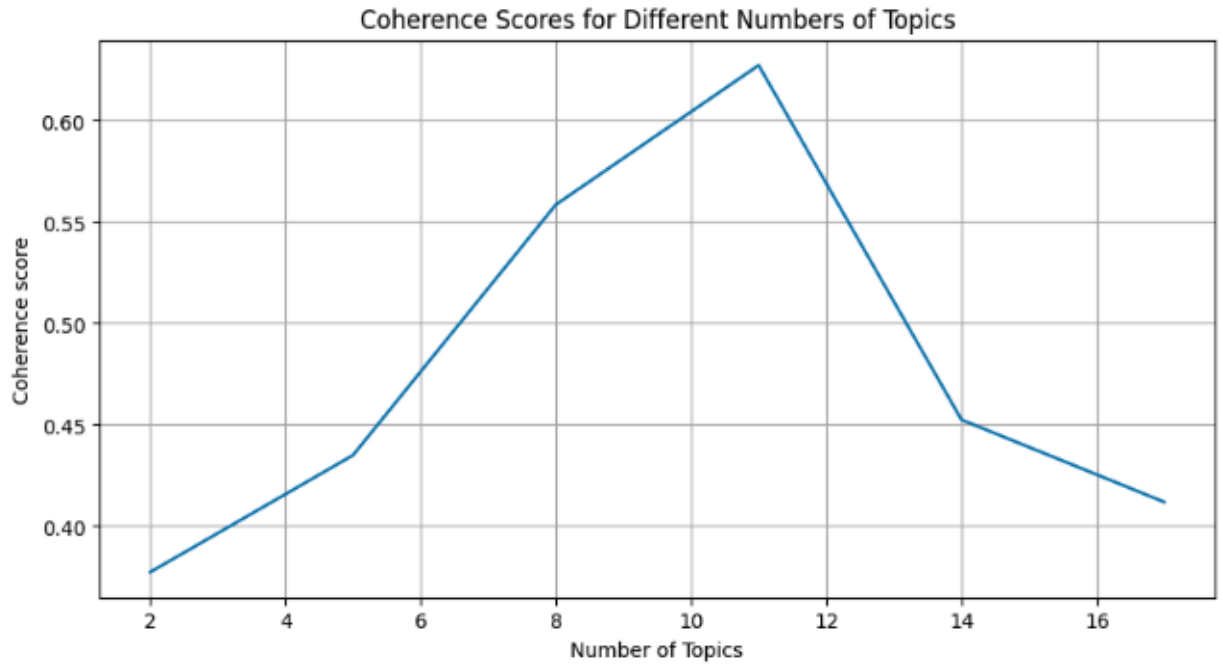


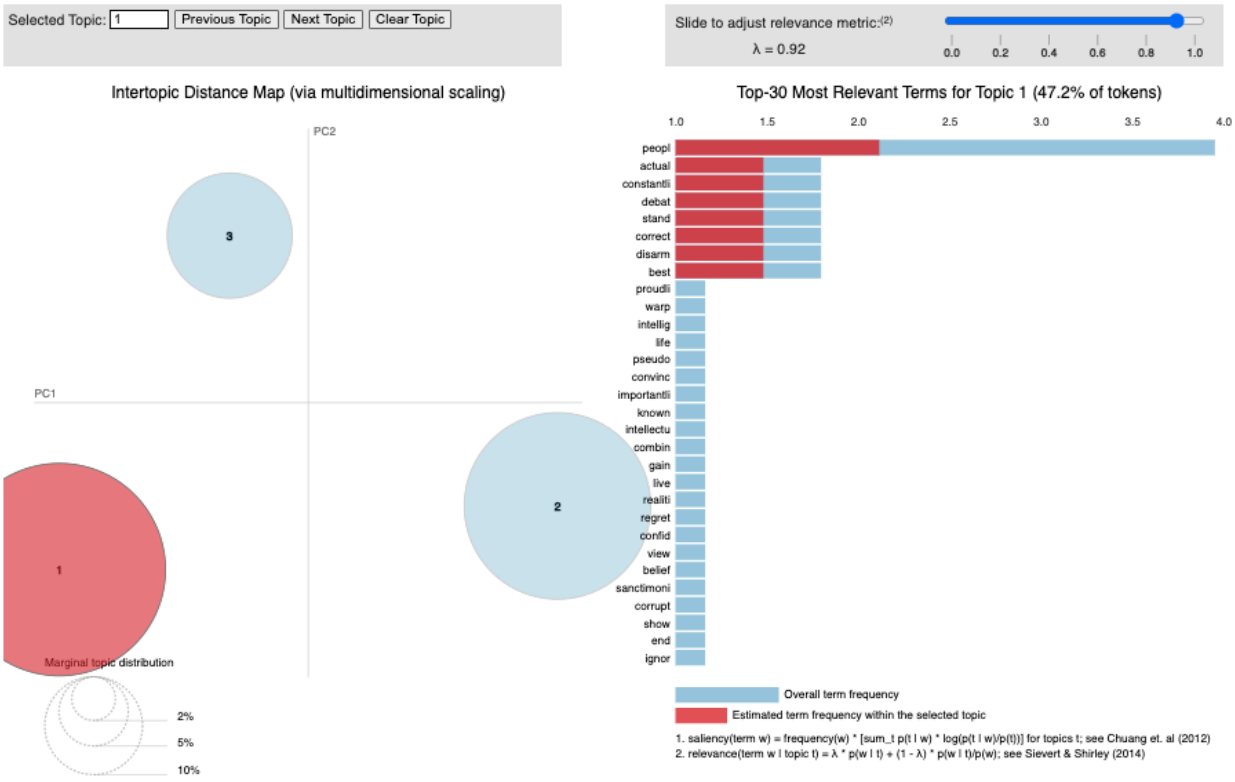
D Static Embedding Training Results

	train_loss	train_accuracy	train_precision	train_recall	train_f1	train_pr_auc	train_roc_auc	val_loss	val_accuracy	val_precision	val_recall	val_f1	val_pr_auc	val_roc_auc	trial_number	value
2	0.000308	0.5091	0.493175	0.454620	0.473114	0.606098	0.507493	0.732141	0.511338	0.500000	0.002275	0.004529	0.494912	0.500050	2	0.654900
0	0.000271	0.5149	0.499588	0.375619	0.428824	0.588954	0.510791	0.694881	0.495331	0.489044	0.731119	0.586069	0.675778	0.500560	0	0.586100
9	0.000041	0.7050	0.707386	0.667698	0.686969	0.768092	0.703899	0.728699	0.504669	0.492697	0.460419	0.476011	0.608394	0.503688	9	0.551200
3	0.000136	0.7460	0.748707	0.716584	0.732293	0.801346	0.745132	0.760473	0.514229	0.503166	0.469973	0.486003	0.616071	0.513247	3	0.527200
7	0.000233	0.7050	0.703648	0.676361	0.689735	0.768455	0.704155	0.754100	0.505780	0.494525	0.513649	0.503905	0.622917	0.505955	7	0.444005

E Topic Modeling

E.1 LDA for False Positives





E.2 LDA for False Negatives