INST750 Assignment 1

Yuanyou Yao

February 1, 2024

Introduction

Unwanted text messages and emails have become a persistent challenge in digital communication. Many individuals, including myself, instinctively check messages upon receiving them, only to find that they are spam. This not only leads to a sense of deception but also wastes valuable time. Similarly, promotional and unsolicited emails clutter inboxes, making it difficult to identify important messages. The process of unsubscribing from these emails is often tedious, requiring manual effort for each sender. For me, protecting personal information and financial security is a top priority in digital communication. Fraudulent text messages containing malicious links pose a significant risk, potentially leading to financial loss and data breaches. Unintentional engagement with such messages can compromise sensitive information, highlighting the need for robust security measures to detect and filter deceptive content. Given these challenges, leveraging cloud-based solutions for filtering and managing unwanted communications could provide a more efficient and effective approach.

For this, I find the dataset of SMS spam detection on Kaggle. It contains one set of SMS messages in English of 5,572 messages, tagged according to being ham (legitimate, not spam) or spam. I build a prediction model that will accurately classify text.

The chosen dataset meets three key criteria that make it well-suited for this study. First, its moderate size ensures that computational demands remain manageable, allowing the SVM model to process the data efficiently. This has been confirmed through actual model training, where calculation times remained within acceptable limits. Second, as a Kaggle dataset, it is well-structured and comprehensive, facilitating reliable analysis. In addition, its use enables comparisons with models developed by machine learning professionals worldwide, providing opportunities for further optimization and improvement.

Methodology

First of all, I did EDA. The dataset shapes `(5572, 5)`, which looks like this:

index	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
95	spam	Your free ringtone is waiting to be collected. Simply text the password \MIX\" to 85069 to verify. Get Usher and Britney. FML	PO Box 5249	MK17 92H. 450Ppw 16"	NaN
281	ham	\Wen u miss someone	the person is definitely special for u..... But if the person is so special	why to miss them	just Keep-in-touch\" gdeve.."
444	ham	\HEY HEY WERETHE MONKEESPEOPLE SAY WE MONKEYAROUND! HOWDY GORGEOUS	HOWU DOIN? FOUNDURSELF A JOBYET SAUSAGE?LOVE JEN XXX\""	NaN	NaN
671	spam	SMS. ac sun0819 posts HELLO:\You seem cool	wanted to say hi. HI!!!\" Stop? Send STOP to 62468"	NaN	NaN
710	ham	Height of Confidence: All the Aeronautics professors wer calld & they wer askd 2 sit in an aeroplane. Aftr they sat they wer told dat the plane ws made by their students. Dey all hurried out of d plane.. Bt only 1 didnt move... He said:\if it is made by my students	this wont even start..... Datz confidence.."	NaN	NaN

Figure 1: `df.head()`

The last 3 columns also contain some meaningful words and sentences, so I decided to merge the 4 columns into a long sentence. And here is the result after transform:

	label	text
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

Figure 2: `df.describe()`

As expected, the data is highly **IMBALANCED**! But I will deal with this issue later.

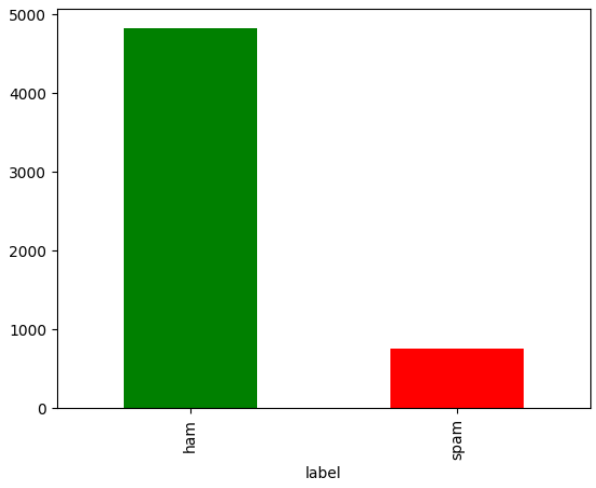


Figure 3: Label Distribution

I labeled `ham = 0` and `spam = 1`. The distribution of `train_features`, `val_features` and `test_features` is `((3930,), (806,), (836,))`, roughly `.7 : .15 : .15`.

As for model selections, I initially chose **Support Vector Machine** and tuned its hyperparameters for better model performance. Compared to logistic regression, decision

trees and random forests, SVM can handle high-dimensional variables and imbalanced data better. Per the assignment requests, I fine-tuned a **RoBERTa** language model. Finally, I tried deep learning **LSTM** and compare three models.

In the evaluation phase, many people care about **false positives**. They give a reason that if timely important business emails or messages were tagged "spam", the circumstance would be much worse. However, from my point of view, people are more aware of looking in the spam folder when they are waiting for important information. I prioritize **recall** because spam message contains links that will steal personal information. If fraudulent text messages are not detected, there is an additional risk of personal information being leaked. Therefore, **recall** is used as the metrics to evaluate and tune models performances.

Models

Support Vector Machine

To deal with the imbalanced issue. I used SMOTE to mitigate it. Initially, I chose `C = 1`, `kerner = 'linear'`, and `class_weight='balanced'`. For the hyperparameter tuning phase, I tried 20 regularization parameters in $[0.1, 2]$ with three different kernels `['rbf', 'poly', 'sigmoid']`. Given the ratio `Counter({0: 3403, 1: 1020})` was updated by SMOTE, I tried class weight `[{0:1, 1:3}, 'balanced']`. The optimized SVM model is:

`(SVC(C=0.6, class_weight='balanced', kernel='sigmoid')`, with `recall = 0.94`

Below is the performance comparison of the intial and final SVMs:

Model Prediction on Test Set		
	Initial	Final
precision	0.99	0.95
recall	0.88	0.90
f1	0.93	0.93
roc_auc	0.98495	0.98443

Confusion Matrix			
Initial		Predicted	
Labels		0	1
Actual	0	723	1
	1	13	99

Confusion Matrix			
Final		Predicted	
Labels		0	1
Actual	0	719	5
	1	11	101

It can be seen that, after hyperparameter tuning, the performance of the SVM model has improved to a certain extent. In general, the recall level is .90, the f1 score is .93. The model prediction ability is good. Because I over-pursued recall, the precision and accuracy of the model decreased, which is also reflected in the ROC_AUC score. By analyzing the confusion matrix, we can see that although two more spam messages were identified, the price paid was that four normal text messages were classified as spam. Therefore,

using the trade-off f1 score may beetter metrics to better balance false positives and false negatives.

RoBERTa

To deal with the imbalanced issue. I weighted the sample using the inverse of frequency. In the loss function, class weights are taken into account. In addition, I set `batch_size=16` for the batch process, `weight_decay=0.001` and a small learning rate (2×10^{-5}) for regularization. A `scheduler` is also used to fine-tune the model more effectively. Below is a runtime screenshot:

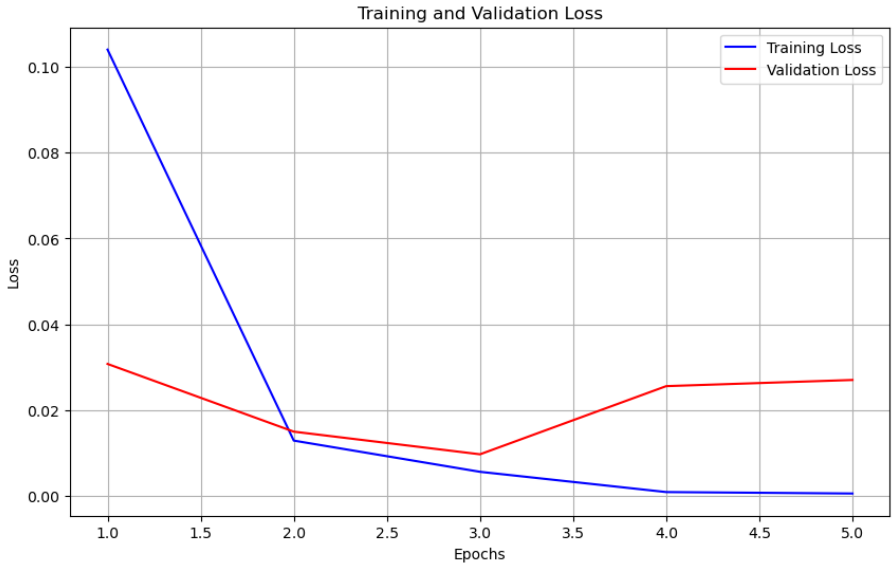


Figure 4: RoBERTa `epochs=5`

I chose the best model from epoch 3 and predict on test set, here is the classification report:

Confusion Matrix			
RoBERTa		Predicted	
Labels		0	1
Actual	0	723	1
	1	2	110

Classification Report				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	724
1	0.99	0.98	0.99	112
accuracy			1.00	836

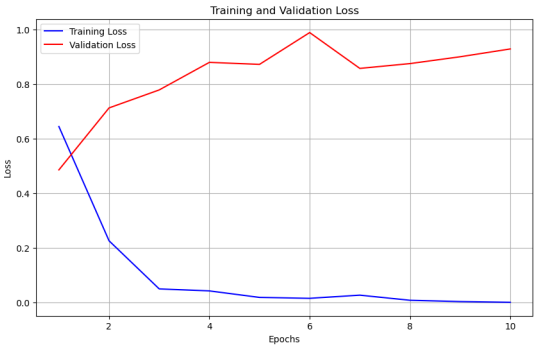
The RoBERTa model provides a powerful detection mechanism with a $ROC\_AUC = 100$.

Long Short-Term Memory

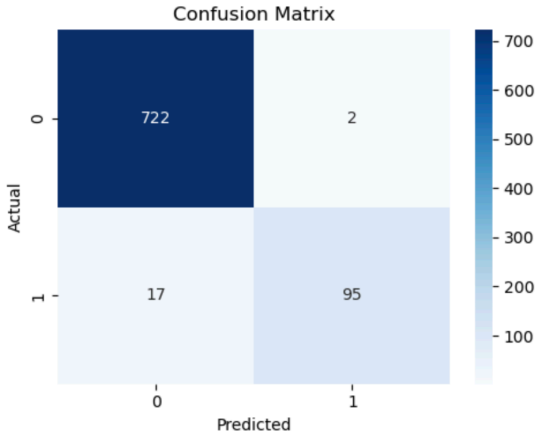
I also built a LSTM model, which learned embedding from sctrach. The hyperparameters are `EMBEDDING_DIM=1000, N_LAYERS=3, DROPOUT=0.5, BATCH_SIZE=16` and

`LEARNING_RATE=0.001`.

The loss function increases from as the epoch goes. So I chose the model from epoch 1. After prediction, the $recall = 0.85$ and $ROC\_AUC = 0.97$. Below is the detail:



(a) Training Phase `epochs=10`



(b) Prediction CM

Figure 5: LSTM Metrics

The LSTM model tends to overfit the training set.

Results

Here is the table of three models metrics. These models represent different levels of complexity and are chosen to evaluate their suitability for classifying the dataset. Overall, RoBERTa outperforms other two models. LSTM has good precision, meaning it won't misclassify legitimate messages. However, its ability of identifying actual positives(spam) messages is worse than SVM. Comparing to advanced models, SVM performance is not enough.

Model Comparison			
Metrics	SVM	RoBERTa	LSTM
precision	0.95	0.99	0.98
recall	0.90	0.98	0.85
f1	0.93	0.99	0.91
roc_auc	0.98	1.00	0.97

Discussion

The first thing I would like to mention is that CUDA is not friendly to `scikit-learn`,

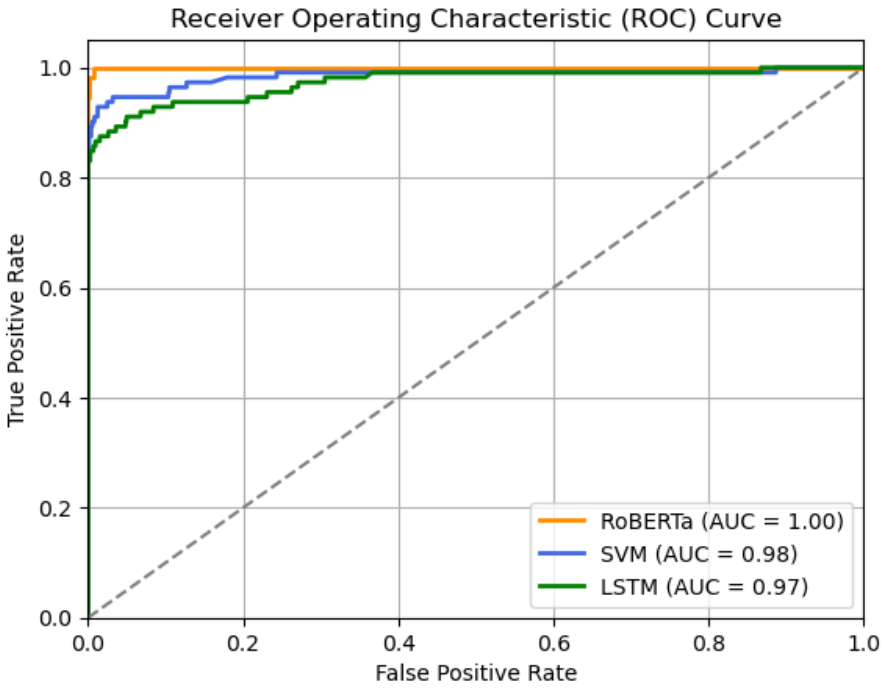


Figure 6: Model ROC Curve

at least for SVM. In fact, it took me less than 3 minutes to train RoBERTa, while 6 minutes for SVM on CPU. I did some research, there is a module `thundersvm` that support CUDA. However, on my end, GPU-boosted Transformers wins efficiency round.

In SVM, Precision and ROC_AUC score drop comparing the first and final models. But I chose the newer model because f1 scores are still the same and prevention ability(recall) improves. However, When it comes to LSTM, there was one epoch, the recall was the optimum but the loss continued going up. The model has overfitting risk. So I decided to make a trade-off by not updating the best model.

The training speed and the evaluation results surprise me as well. On the one hand, the small dataset makes it easy to train. On the other hand, the dataset may have similar pattern, which means the model may be overfitted. When the size goes larger, the prediction ability decreases. So for further study, I would like to use other sources to test models robustness.

References

I leveraged different GenAI tools, including ChatGPT, Claude. With the help of these AI tools, am I able to write Python Classes and functions. I also referred to 2 Kaggle notebooks, Christian and Pablov. Lastly, I refer to CSDN for full understanding of concepts covered during lectures and other related state-of-the-art technologies.