AI Machine-Learning & Analytics
Deep Learning Project

BUSINESS SCHOOL

Scam/Spam SMS Generation and Detection

Group 4
Matias Arevalo
Pilar Guerrero
Moritz Goebbels
Tomás Lock
Allan Stalker

Table of Contents

1. Executive Summary

This report details a project investigating the escalating threat of the misuse of AI that generates harmful, malicious or unwanted messages (spam) and the development of robust detection methods. It demonstrates that while AI enables more sophisticated scams and the ability to generate at volume, AI-based detectors still can offer significant protection. The core problem addressed is the increasing use of accessible Large Language Models (LLMs) by malicious actors to create novel, convincing scam messages at scale, overwhelming typical keyword and volume-based filtering systems. To address this problem, we developed a dual-framework simulating both attacker and defender roles using fine-tuned LLMs, which yielded near-perfect detection results with a RoBERTa-based classifier. This framework powers "Scam Busters", our proposed end-user product for scam message detection. Our study demonstrates the effectiveness of responsibly designed AI in decreasing digital risks and improving trust in messaging services.

2. Problem Statement

The rise of large language models (LLMs) like ChatGPT has democratised content generation at scale. While these tools empower users, they also enable a new class of adversaries. Scammers can now generate highly persuasive and fraudulent messages at minimal cost and higher volume. Unlike traditional scams, which featured repetitive keywords, poor grammar, and mass distribution, AI-generated scams are fluent, varied, and tailored to bypass keyword filters and volume-based detection systems. These scams blend in with legitimate communication, making them significantly harder to detect using conventional heuristics. This growing threat is already having real world consequences: recent studies estimate that AI-enhanced scams caused over $1 trillion in losses in 2024 alone (Google, 2024). Yet academic and industrial spam filters are often reactive, designed around legacy patterns rather than proactive detection of AI-native threats.

3. Originality & Impact

3.1 Originality

This project's originality lies in its cat & mouse dynamic, acting as both attacker and defender to realistically model and counter the evolving threat landscape, technically implemented through an adversarial training loop. On the attacker side, it relies on a fine tuned LLaMA-3 model to generate realistic scam messages using reverse engineered prompts from real scams via a custom prompt mapping API. While on the defensive side, we fine-tuned a RoBERTa-based transformer model, which is capable of detecting both real and synthetic spam messages with high precision. Furthermore, besides the technical depth this project targets a critical and underserved niche, the SMS scam detection market. Finally, the product is especially focussed on building trust with end users by classifying scam messages through a user-friendly interface.

3.2 Impact

This project showcases an applied approach in assessing and strengthening the AI scam content mitigation. While communication is primarily only being done through mobile devices, the volume of harmful messages and the sophistication behind them has increased rapidly. This project seeks to fill that gap by employing generative models to simulate real-world attacker behavior and testing detection systems. This shifts detection from pattern-matching filters to semantic-understanding models such as RoBERTa, which highlights the need to advance detection systems toward greater understanding.
Finally and most importantly, the project highlights the role of responsible AI development. Understanding the technologies' dual-uses, we focus on the malicious side by constructing language generation models meant for resilience and ethical safeguards. Thus, the project serves both academic research and real world application, offering a tool to decrease user risk and enhance trust in platforms.

4. Market Analysis

The growth of the anti-spam and fraud detection industry correlates with the ever-evolving digital threats and advancements of AI technology. In 2023, an estimated 347 billion emails were sent daily (Statista, 2024), of which nearly half were spam. While email systems are well protected, SMS and social media platforms lack sufficient measures.

4.1 Market Size and Key Trends

The global fraud detection and prevention market was valued at $33.1 billion in 2024 and is expected to reach $60 to $100 billion by the end of the decade (Grand View Research, 2024). Moreover, the messaging security market is expected to reach a market size of approximately $10 billion by 2025 and further grow to $20.5 billion by 2029 at a CAGR of 18.9% (The Business Research Company, 2024). Email security represents a fundamental part of this market, with the business email security segment approximately worth $7–8 billion in 2023 with double-digit growth (Fortune Business Insights, 2024). Additionally, the SMS firewall market, though a smaller niche, is evolving as operators around the globe install or upgrade firewalls; industry reports indicate telecom spending driven by regulatory mandates aimed at curbing SMS fraud. Geographically, as of now North America is historically the largest market for messaging security because of high enterprise adoption and spam volume, while the Asia-Pacific region is the fastest growing due to the surge in smartphones, digital services, and in some areas an avalanche of SMS spam that has led to substantial anti-spam infrastructure spending by both carriers and governments.

4.2 Competitor Landscape

Competitor Type	Category	Examples	Role / Focus
Direct	Email Security	● Google (Gmail/RETVec) ● Microsoft(Outlook/Exchange) ● Proofpoint ● Mimecast ● Cisco Secure Email ● Barracuda ● Apache SpamAssassin (Open Source)	● Filtering spam/phishing in consumer/enterprise email ● Advanced threat protection (sandboxing, URL analysis)
Direct	SMS/Messaging	● Carrier Firewalls (Tata Communications ● Infobip, Mobileum) ● Twilio ● Google Messages (on-device AI) ● Apple Messages ● Truecaller	● Network-level SMS spam blocking (carriers) ● API compliance (CPaaS) ● Device-level scam warnings.
Indirect	Social Platforms	● Meta (Facebook/Instagram) ● X (Twitter) ● LinkedIn	● Internal moderation systems using AI/human review to remove bots, fake accounts, spam posts/DMs.
Indirect	Enterprise Fraud	● NICE Actimize ● Feedzai ● FICO Falcon ● SAS Fraud Management ● Arkose Labs ● HUMAN Security	● Detecting financial fraud, account takeover, bot activity ● Ingesting communication signals to provide context.

Figure 1: *Competitor Analysis*

4.3 Strategic Positioning

Most current solutions tend to operate at an enterprise level, while our project is positioned to serve end-users. Unlike email security gateways or embedded messaging filters, we imagine a user-friendly application, like a WhatsApp chatbot, in which users can check whether a message is a scam or safe. Current firewalls and AI detection systems lack user engagement transparency and are only accessible to large organizations, but Scam Busters will let anyone verify messages without technical expertise. This allows us to build digital trust and user confidence among vulnerable demographics such as elderly users, non-technical consumers, or those in high scam regions.

5. Viability Analysis

5.1 Technical Feasibility

Our project has already demonstrated a high level of technical feasibility which is proven with our whatsapp based MVP that was built using the Twilio service. It demonstrates a commercially deployable service in the framework of a chatbot, though this does come with some compromises in user convenience. If we were to directly integrate our product instead of a simple chatbot, there would be a significant privacy trade off as our model would have to read every message that goes to the user on a virtual server as the current size of the model is far bigger than any smart mobile device can store. Given this, we have come up with two different approaches to enable a more sophisticated and compliant commercial deployment:

<u>Model distillation:</u>
This would involve compressing our original model of approximately 7 billion parameters to a smaller distilled model with approximately 100 million parameters. This distilled model could then run locally on a user's device without needing messages to be sent anywhere. However, this would introduce a technical trade off with a slight reduction in detection accuracy that would be approximately 3-5% lower than the original model. It may also strain the device due to the processing effort required.

<u>Pre-Filtering Pipeline:</u>
The second alternative involves implementing a lightweight, rule-based pre-filter directly within the messaging app, assessing incoming messages based on simple yet effective criteria (unknown senders, suspicious links, specific scam-related keywords). Only messages identified as potentially suspicious by this pre-filter are then securely forwarded. This approach would significantly reduce the user's compromise on privacy as only 3% of their messages would be sent externally for further inspection. It also drastically cuts down computational costs and latency by processing the vast majority of "safe" messages locally.

From a technical standpoint, the second option seems to be more feasible for a commercial deployment that respects the privacy and security of the user and complies with regulations.

5.2 Economic Feasibility

The prototype was developed with minimal cost using easily accessible tools:

- Model training and experimentation using Google Colab & Google Colab Pro+.
- Focusing on LLM fine-tuning for A100 GPUs with Unsloth.
- Open access libraries such as Hugging Face Transformers, Pytorch, and Scikit-learn.
- Deploying the MVP with the Twilio WhatsApp API sandbox environment.

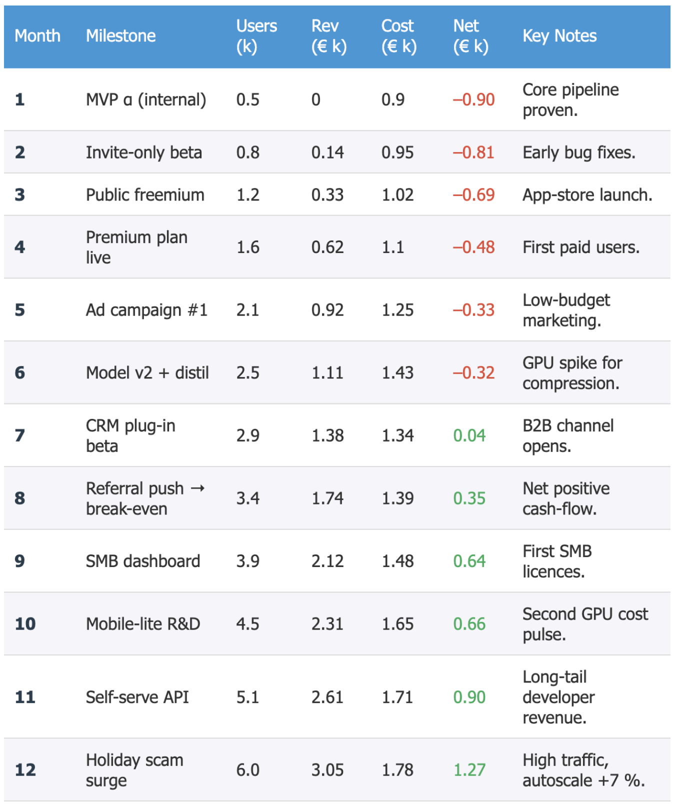
Future costs for expansion would include:

- Approximately €50-€100/month for cloud hosting (model inference endpoints).

- For increased program usage, additional €10/month per 500-1,000 new users (estimated baseline).
- Prompt mapping API: €460 per 1m examples (0.003/k tokens)
- Generator & detector training €480 per cycle - 8 x A100 @ €2.5 hr x 24 h
- Storage monitoring and backups: €100-150 monthly

5.3 Development timeline

In order to reach the target market, we have prepared a 12-month rollout plan:



Month	Milestone	Users (k)	Rev (€ k)	Cost (€ k)	Net (€ k)	Key Notes
1	MVP α (internal)	0.5	0	0.9	−0.90	Core pipeline proven.
2	Invite-only beta	0.8	0.14	0.95	−0.81	Early bug fixes.
3	Public freemium	1.2	0.33	1.02	−0.69	App-store launch.
4	Premium plan live	1.6	0.62	1.1	−0.48	First paid users.
5	Ad campaign #1	2.1	0.92	1.25	−0.33	Low-budget marketing.
6	Model v2 + distil	2.5	1.11	1.43	−0.32	GPU spike for compression.
7	CRM plug-in beta	2.9	1.38	1.34	0.04	B2B channel opens.
8	Referral push → break-even	3.4	1.74	1.39	0.35	Net positive cash-flow.
9	SMB dashboard	3.9	2.12	1.48	0.64	First SMB licences.
10	Mobile-lite R&D	4.5	2.31	1.65	0.66	Second GPU cost pulse.
11	Self-serve API	5.1	2.61	1.71	0.90	Long-tail developer revenue.
12	Holiday scam surge	6.0	3.05	1.78	1.27	High traffic, autoscale +7 %.

Figure 2: *12-month rollout plan*

This rollout plan outlines a preliminary roadmap of how our product could evolve commercially and the kind of sectors we could potentially serve beyond B2C. The first quarter would focus on product validation and monetisation through freemium and premium users tiers. After the halfway mark we could introduce a CRM plug-in to enable our spam detection tool to integrate into customer relationship management systems such as hubspot. By the ninth month we may introduce a dashboard for small and medium sized businesses for internal use in smaller companies that provides a layer of security for them. Finally, in month 11, we could potentially launch a self-serve API that allows external developers or businesses to integrate our detection model independently into their systems, creating a long-tail revenue stream.

6. Technical Solution

6.1 Data Requirements and Sources

We explored the internet for scam messages and found the following reliable datasets. We even tried generating some with AI but these appeared to be very deterministic with predefined patterns always ending with a link and not very valuable. By merging several datasets and filtering out duplicates, given that some datasets where similar or where already a combination of others we generated our own containing 28749 observations. Our sources include the following. Refer to the ***Appendix Figure 1 & Text 1*** for Data sources and data exploration.

6.2 Data Pipeline

The data processing flow involved several stages, visualised in the figure below

1. Ingestion: Collection of real-world spam/scam SMS messages.
2. Prompt Mapping with custom API.
3. Generator Fine-Tuning: Llama-3 8B model with prompt pairs.
4. Synthetic Data Generation
5. Detector Training Dataset: Combining the original messages with the newly synthetic spam messages.
6. Detector Training: Fine-Tune RoBERTa

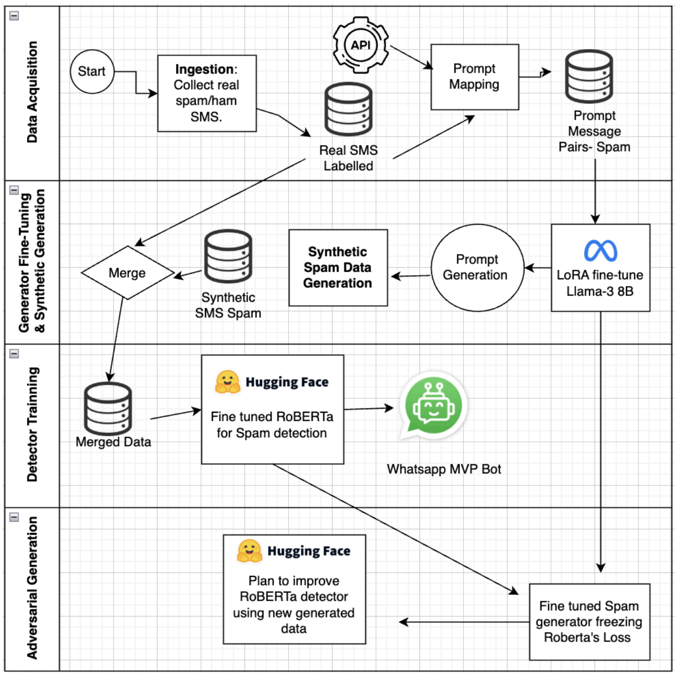

Data Acquisition
Start
Ingestion: Collect real spam/ham SMS.
API
Real SMS Labelled
Prompt Mapping
Prompt Message Pairs- Spam
Generator Fine-Tuning & Synthetic Generation
Merge
Synthetic SMS Spam
Synthetic Spam Data Generation
Prompt Generation
LoRA fine-tune Llama-3 8B
Detector Trainning
Merged Data
Hugging Face
Fine tuned RoBERTa for Spam detection
Whatsapp MVP Bot
Adversarial Generation
Hugging Face
Plan to improve RoBERTa detector using new generated data
Fine tuned Spam generator freezing Roberta's Loss

Figure 3: Data Pipeline

6.3 Scam Generation Model Process

Upon our first completion of baseline models we achieved surprisingly good results. However, we quickly identified this as stemming from limited and imbalanced training data strongly favoring non-scam messages. To address this, we decided to fine-tune large language models (LLMs) to generate additional data.

Our initial attempt involved fine-tuning GPT-2. Human evaluation revealed outputs lacked realism, often missing context and purpose. For example, one output was, “spammer to unsubscribe from this list”. Realizing we needed a more powerful model, we moved to Mistral’s open-source model. Despite Mistral’s strong architecture and 7 billion parameters, we encountered major practical issues. Checkpoints were challenging to load correctly, outputs remained incoherent, and extensive troubleshooting provided no guarantee of eventual success. For these reasons, we pivoted again.

Next, we explored LLaMA-2, a highly suitable model, but it failed due to memory constraints. To overcome this, we adopted the Unsloth framework. This allowed us to successfully fine-tune a LLaMA-2 7B model using 4-bit quantization, significantly reducing memory usage without a performance loss.

Initially we fine-tuned with a single, generic prompt. This led to repetitive outputs. To diversify our training data, we generated individual prompts for each scam. Manually writing prompts for thousands of messages was impractical, so we automated prompt creation using the OpenAI API. By reverse-engineering prompts from existing scams, we significantly increased the variety and realism of generated scams.

Finally, we integrated these improvements into fine-tuning the LLaMA-3 model, producing clearer, contextually relevant scam messages. To evaluate our generator’s effectiveness, we tested baseline detectors on both original and generated messages. The performance remained largely unchanged, suggesting the synthetic messages closely resembled the original dataset. Manual reviews further supported this, revealing most generated scams as realistic, credible, and diverse. However, as generation quality improved and message volume increased, manual evaluation became more subjective.

A significant limitation is that our generated messages closely mirror the original data. Our detector learned from the same patterns used by the generator, meaning synthetic messages did not introduce genuinely new linguistic challenges. Thus, the model may have artificially strong recall on our test data. Continuous acquisition of new real-world scams will remain crucial for robustly assessing the detector’s true generalization capabilities.

6.4 Scam Detection

Initially we focused on training a detector with our real data sources. The first Baseline used CountVectorizer with a Multinomial Naive Bayes classifier. To explore further we attempted a TF-IDF SVC. This simple model was incredibly successful in classifying spam and ham with an average accuracy of 0.97 and a f1-score of 0.95 on the ham. This primarily showcased the limitations we had and our data being very linearly separable making it fairly easy to detect spam.

Challenged by these findings and aiming to create a substantial project we switched to attempting detecting obfuscated data with the idea that this will further challenge our detector.

Figure A2 in the Appendix summarizes the results of our experiments. These results were much lower than our previous baseline showcasing obfuscated data was in fact challenging the detector. From our baselines, to take it a step further we also did some trials with transformers including DistilBert, RoBERTa and tested different losses. Eventually we seemed to have reached a precision-recall tug of war and guided by our instructor we decided to include a generator model into our project instead. Once available the generated new messages we started by applying the same initial baseline models to our merged data.

Even with the generated data, the SVM was very effective in detecting data with a 97% accuracy. When reflecting on this, it made sense given that the generator had very likely learned the same patterns the SVM was already successful at detecting. Regardless of these high results we decided to take it a step further and finetune RoBERTa, a Transformer-based language model with stacked self attention models that extends BERT for better contextualised representations.

We used RoBERTa's pertained tokenizer and a custom loss overriding the default cross-entropy so that each spam example is penalized twice as heavily as a ham example. Our model included gradient accumulation resulting in the actual batch size being 32. We settled on a low learning rate of 2×10^{-5} to finetune the transformer. We also included a Cosine decay schedule which gradually reduced the LR with 100 warm up stamps. Additionally weight decay was included to discourage excessively large weight as well as gradient clipping to stabilise training and avoid exploding gradients. In terms of regularisation we included early stopping with a patience of 2 in terms of f1 to avoid overfitting.

From training curves available in Appendix Figure A3 we can see how there is a rapid convergence where training loss plummets by epoch 3, with val loss also hitting its lowest with 0.049. After this there is a slight overfitting as validation loss increases even though accuracy and AUC remain incredibly high in the evaluation set. The highest F1-spam score occurs at epoch 10 (0.986) but it is very probable epoch 3-5 would provide identical performance.

Finally we tested RoBERTa spam detector on the test data as seen in Appendix Figure A4. The 99% performance across most of our metrics proved RoBERTa was basically perfect at classifying spam. When scrolling through false positives and false negatives the instances were to our surprise even difficult to classify for a human on top of being only 1% of our data.

6.5 Adversarial Generator

At this point, we wanted to see if an adversarial approach could benefit both our generation and detection models. Inspired by the GAN framework, we implemented a custom training loop where a language model was trained to generate spam-like messages to bypass our fine-tuned RoBERTa detector.

We used a form of curriculum learning in which the generator was progressively exposed to increasingly difficult detectors to improve its evasive abilities, starting with simpler models and ending with RoBERTa, our most robust detector made in the stage before.

The first detector we tried was Naive Bayes, one of the simplest models from earlier experiments. When we trained our fine-tuned LLaMA-based generator against it, the new outputs were more successful at fooling RoBERTa than the original generator. However, this model's evasion skills came at the cost of fluency, as the messages were often unnatural and sometimes too technical, which were clear spam to the naked eye. To fix this, we retrained the generator with key adjustments to the loss function aiming to balance effectiveness and fluency.

These changes significantly improved output quality: the messages sounded more human made but still contained spammy content, and the evasion rate stayed at similar levels. We saved this checkpoint as a stronger foundation for the next stages.

With this improved generator, we moved on to more complex detectors using TF-IDF + Logistic Regression models with unigrams, bigrams, and trigrams. At each step, we retrained the generator using the same adversarial loop.

Surprisingly, the bigram-based TF-IDF model led to the best generator model against RoBERTa. Even though it was slightly less evasive than other setups, it produced messages that were noticeably more fluent and realistic, which could potentially trick the human eye.

With these findings, even though the improvement was only around 2%, we were able to show that implementing a curriculum-based adversarial training framework can improve both evasion and fluency in spam generation, and could be further improved with a more diverse dataset.

7. Future Development Path

While our prototype is strong and performs tasks as intended it is not yet a final deployable product. Due to data limitations discussed, for a final product, we would be collaborating with a large messaging platform such as WhatsApp or Messages who have the large and quality datasets with scam messages which we would be able to utilise their data in order to generate stronger scam detecting models.

Moreover, a direct integration into the messaging platform domain rather than a chatbot. Additionally, the model gets tuned with the messages of the users to understand how the users communicate to have a better performance. Applying transfer learning using some historical data of messages from past conversations provided the user consents to this and classifies the messages we intended on using.

Regarding the implementation, the detector only intervenes if it detects a scam. A lightweight version of the model integrated into the messaging platforms verifying if each message received is a scam. Upon detection, passing a certain threshold, the message would be run on the complete detector model to verify if it is a scam. If it is, the user will be notified and asked to verify before any further action is taken with the potential scammer. We would provide guides supporting users through the verification process to ensure they do not get scammed. At the end the user would have the option to update us, confirming if the message was a scam or not, or reporting any scams which the model did not pick up. This serves as manual labeling, allowing us to collect real-world feedback and continuously improve the model's performance over time as the scams get more elaborate. In short, the user would be able to use their messaging platform normally and we would only come in contact with the user if there is a potential threat of a scam message.

8. Conclusion

The main limitation we have had over the course of the project was the collection of data which has restricted the potential of our models. The collection has been limited primarily by privacy reasons. Messages datasets contain sensitive information, so sharing messages raises significant privacy concerns and may violate data protection regulations limiting the amount of publicly available data. This made us pivot our approach and generate samples ourselves through fine tuned models.

Our project manages to showcase that even with limited data, improvement in detection is possible. By utilising LLMs as fraudsters, and studying their generation process as well as the detection our project managed to improve performance achieving near perfection by exploiting an adversarial setup. This methodology can provide a competitive advantage for commercial deployment and a large potential for a viable product and service.

Bibliography

Fortune Business Insights. (2024). Messaging Security Market Size, Share & Growth Analysis Report, 2024–2032. https://www.fortunebusinessinsights.com/messaging-security-market-109271

Grand View Research. (2024). Fraud Detection and Prevention Market Size, Share & Trends Analysis Report. https://www.grandviewresearch.com/industry-analysis/fraud-detection-prevention-market

Google. (2024, February). AI-enhanced scams and online security trends. Google Online Security Blog. https://security.googleblog.com/2024/02/

Statista. (2024). Daily number of e-mails sent worldwide from 2017 to 2026. https://www.statista.com/statistics/456500/daily-number-of-e-mails-worldwide/

The Business Research Company. (2024). Messaging Security Global Market Report 2024. https://www.thebusinessresearchcompany.com/report/messaging-security-global-market-report

Appendix

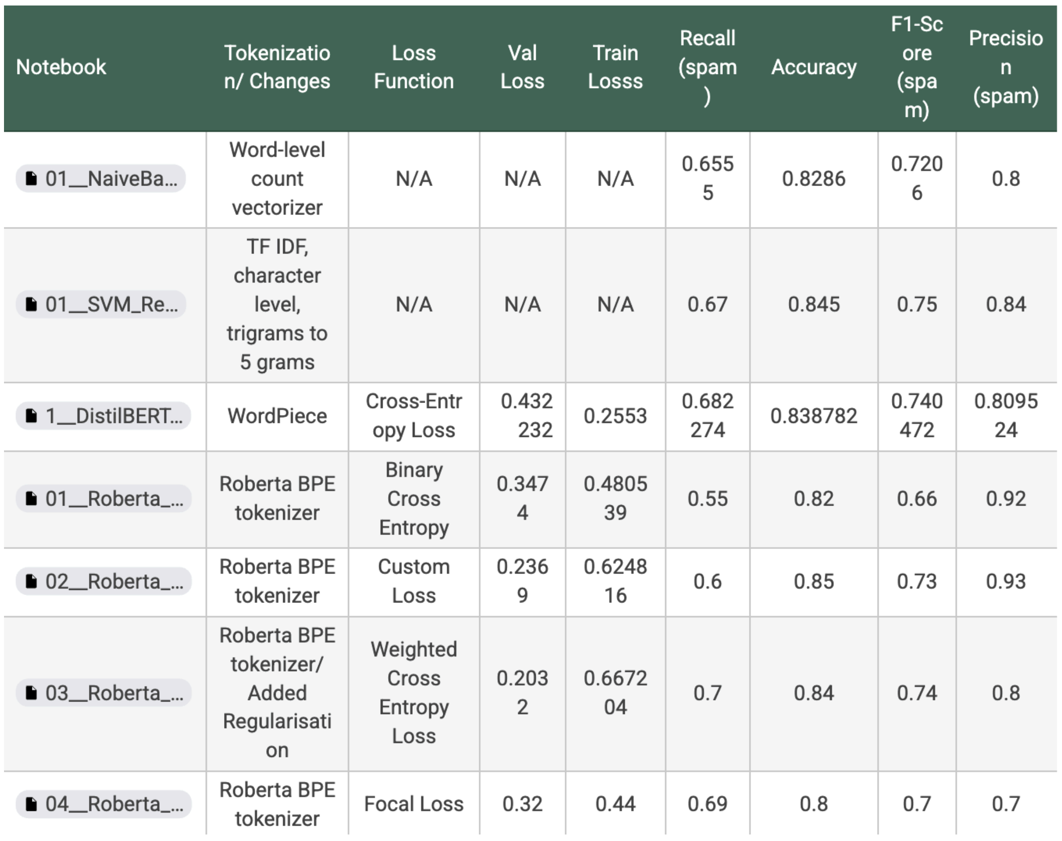
Figure A1: *Dataset sources*

Dataset_Name	Source	Number of Samples
filtered_data	https://data.mendeley.com/datasets/f45bkkt8pr/1	5971
spam[1]	https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset?resource=download	5,169
telegram_spam	https://www.kaggle.com/datasets/mexwell/telegram-spam-or-ham	20,334
sms_spam	https://www.kaggle.com/datasets/dilpreetkaur17/sms-spam-dataset	5,156

Text A1: Exploration of the data
As expected we faced a data imbalance problem with only 28% of our data being spam. This will be addressed further on in the project with generation mechanisms. Moreover figures below give useful insights into the dataset such as the spam word cloud visualising which words are more useful when identifying spam, the character and word count which is fairly balanced across classes as well as the number of special characters and urls with spam as expected having more links. Moreover the last figure shows a sample of the last dataset with preprocessing relevant to the detector applied to replace emojis, phone numbers and links with tokens.

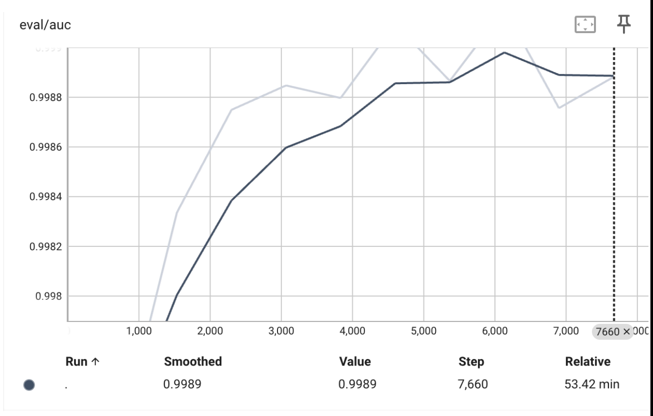
[1] to open this file use: pd.read_csv('spam.csv', encoding='latin-1')

Figure A2: *Model Trials for Obfuscated Data detection*

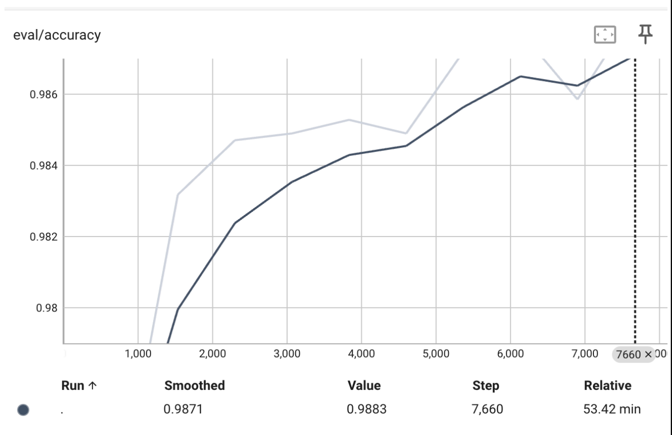


Notebook	Tokenization/ Changes	Loss Function	Val Loss	Train Losss	Recall (spam)	Accuracy	F1-Score (spam)	Precision (spam)
01__NaiveBa...	Word-level count vectorizer	N/A	N/A	N/A	0.6555	0.8286	0.7206	0.8
01__SVM_Re...	TF IDF, character level, trigrams to 5 grams	N/A	N/A	N/A	0.67	0.845	0.75	0.84
1__DistilBERT...	WordPiece	Cross-Entropy Loss	0.432232	0.2553	0.682274	0.838782	0.740472	0.809524
01__Roberta_...	Roberta BPE tokenizer	Binary Cross Entropy	0.3474	0.480539	0.55	0.82	0.66	0.92
02__Roberta_...	Roberta BPE tokenizer	Custom Loss	0.2369	0.624816	0.6	0.85	0.73	0.93
03__Roberta_...	Roberta BPE tokenizer/ Added Regularisation	Weighted Cross Entropy Loss	0.2032	0.667204	0.7	0.84	0.74	0.8
04__Roberta_...	Roberta BPE tokenizer	Focal Loss	0.32	0.44	0.69	0.8	0.7	0.7

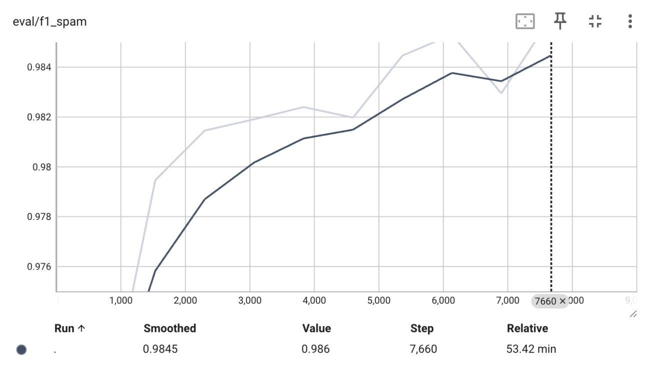
Figure A3: *RoBERTa Training Curves*



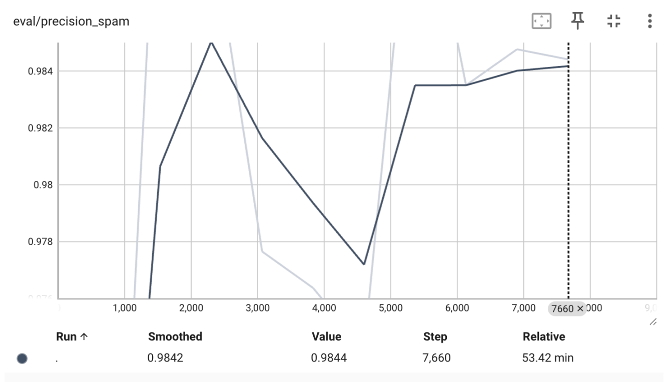
	Run ↑	Smoothed	Value	Step	Relative
●	.	0.9989	0.9989	7,660	53.42 min



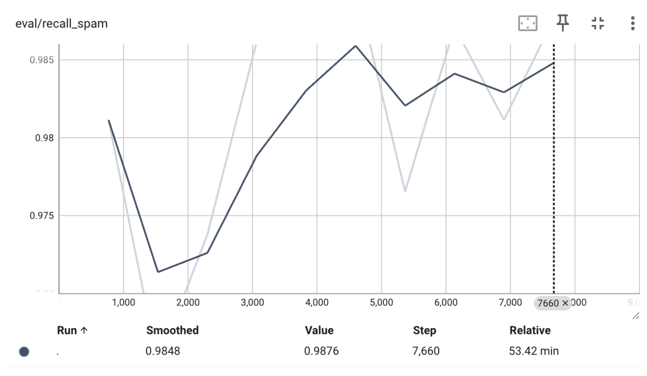
	Run ↑	Smoothed	Value	Step	Relative
●	.	0.9871	0.9883	7,660	53.42 min



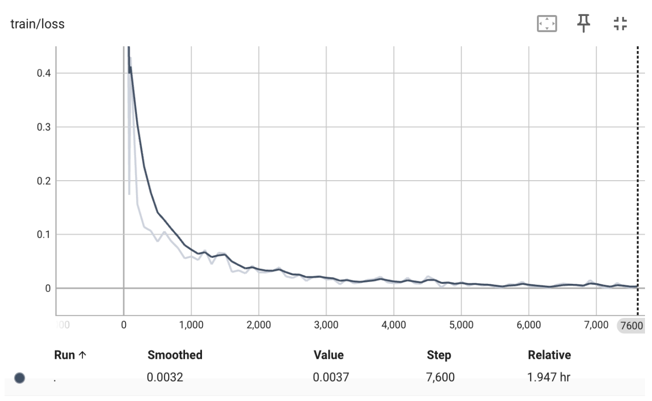
	Run ↑	Smoothed	Value	Step	Relative
●	.	0.9845	0.986	7,660	53.42 min



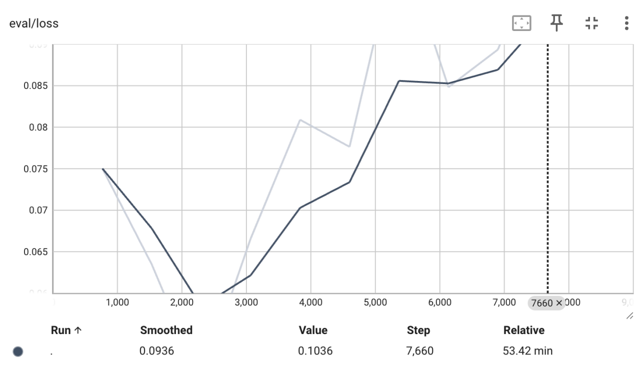
	Run ↑	Smoothed	Value	Step	Relative
●	.	0.9842	0.9844	7,660	53.42 min



	Run ↑	Smoothed	Value	Step	Relative
●	.	0.9848	0.9876	7,660	53.42 min

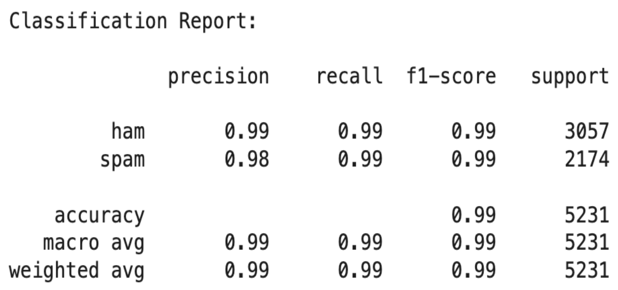


	Run ↑	Smoothed	Value	Step	Relative
●	.	0.0032	0.0037	7,600	1.947 hr



	Run ↑	Smoothed	Value	Step	Relative
●	.	0.0936	0.1036	7,660	53.42 min

Figure A4: *RoBERTa Metrics on Test set*

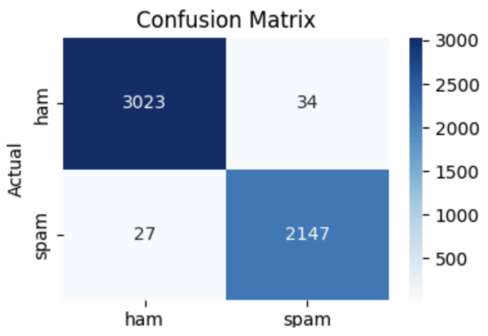


```
Classification Report:

              precision    recall  f1-score   support

         ham       0.99      0.99      0.99      3057
        spam       0.98      0.99      0.99      2174

    accuracy                           0.99      5231
   macro avg       0.99      0.99      0.99      5231
weighted avg       0.99      0.99      0.99      5231
```



Github Repository

The following link can be used to find our project's repository:

https://github.com/Scam-Busters/Spam-Scam-Detection-Model