SmartMail: Integrated Spam Filtering and Automatic Email Response

Siqi CHEN, Ghanibhuti Gogoi, Yuhan CHEN

1. Framework Design

We proposed SmartMail, an end-to-end intelligent email processing framework that integrates spam detection, automated response generation, and a web-based user interface. At the core of the framework lies two fine-tuned language models that have been customized using domain-specific datasets to accurately classify incoming messages and generate contextually appropriate replies.

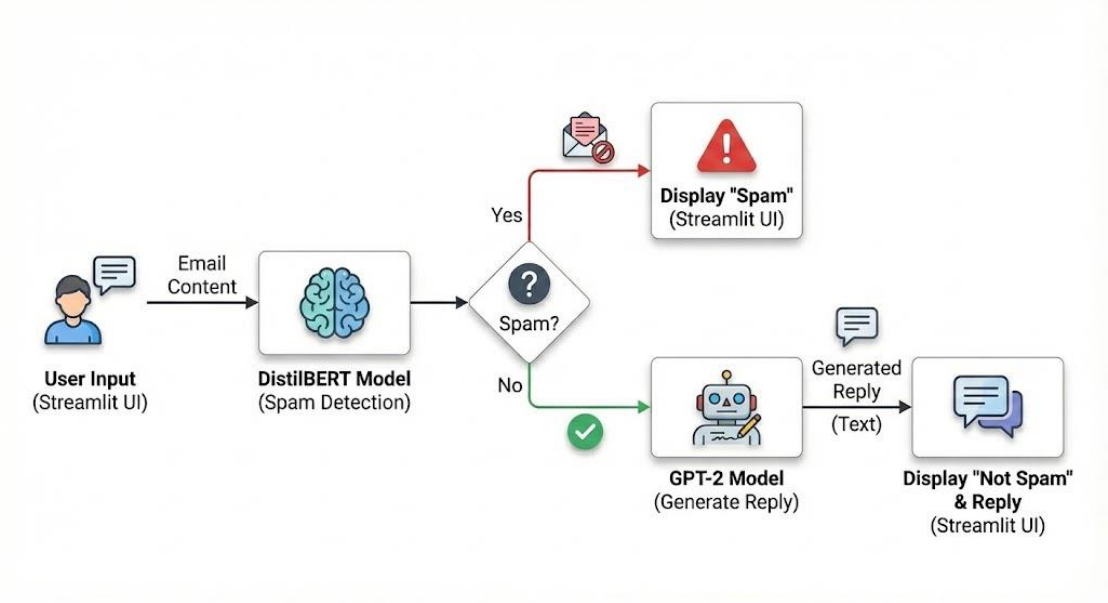

Figure 1: SmartMail framework

As illustrated in **Figure 1**, SmartMail consists of three primary modules:

(1) Spam Detection Module: Incoming emails are passed through a fine-tuned BERT-distilled model that classifies the emails into spam or legitimate ones. This module is optimized for high precision to minimize false positives and ensure that valid messages are not incorrectly filtered.

(2) Auto-Reply Generation Module: Emails that are classified as spam are blocked, and their replies will not be generated. For those classified as legitimate, a fine-tuned GPT-2 produces corresponding responses. This module is optimized for the fluency, relevance, and politeness of the generated replies.

(3) Web Front-End Module: A user-friendly web interface (**Figure 6** in the **Appendix**) displays filtered emails, system predictions, and the generated responses. Users can review, edit, or approve replies before sending, ensuring human oversight and maintaining communication quality.

2. Methodology

2.1. Dataset

To support both spam classification and email reply generation tasks, we utilize multiple publicly available email corpora. For the **spam classification task**, two benchmark datasets are employed: the **SpamAssassin dataset** and the **EnronSpam dataset**. These datasets provide labeled email samples covering diverse spam categories, enabling robust supervised training. For the **reply generation task**, we extract all *non-spam* email samples from the EnronSpam dataset. Each selected email is used as input to the *neuralchat-7b* model running in Ollama with the following prompt:

"Write a short, professional email reply to this message. Output ONLY the reply body. Do not include subject lines or placeholders. {Email}"

This procedure constructs a high-quality synthetic parallel corpus, referred to as the **Synthetic Reply Dataset**, in which each email is paired with a model-generated professional reply. This dataset is subsequently used to fine-tune GPT-2 for supervised email reply generation.

For final performance measurement, we adopt the **SpamMails Dataset** as the unified **held-out test set**. This dataset is not used during training and provides an unbiased basis for evaluating both classification accuracy and reply quality.

2.2. Training

2.2.1. Spam Classification Model Training

We fine-tune DistilBERT for binary spam detection. The datasets are preprocessed into text-label pairs and fed into a HuggingFace Transformers training pipeline. The model is optimized to predict whether an incoming message belongs to the *spam* or *ham* (non-spam) category. Training runs on two NVIDIA T4 GPUs, enabling efficient mini-batch optimization. The final model is saved and later used as the first-stage classifier in the SmartMail framework.

2.2.2. Reply Generation Model Training

For the generative component, we fine-tune GPT-2 using the Synthetic Reply Dataset. This supervised fine-tuning setup enables GPT-2 to learn stylistically consistent, concise, and professional email responses. Training is also executed on dual T4 GPUs, allowing stable training with realistic email lengths. The resulting model serves as the system's reply generator.

2.3. Evaluation

The evaluation procedure consists of two components: (1) objective spam classification metrics and (2) LLM-based scoring of generated replies.

2.3.1. Spam Classification Evaluation

The fine-tuned DistilBERT classifier is evaluated on the SpamMails test set. Predictions are generated in batches and compared against ground-truth labels. We compute *Accuracy* and *Macro-averaged F1 score*. These metrics quantify the model's ability to correctly identify both spam and non-spam messages.

2.3.2. Reply Generation Evaluation

For reply generation assessment, we first select the ham emails in the test set. The fine-tuned GPT-2 model generates a reply for each message using nucleus sampling (top-p = 0.9) and temperature-controlled decoding.

To measure reply quality, we use an LLM-based scoring mechanism with Qwen3-14B. For every generated reply, we construct a scoring prompt (Appendix 1) and request a JSON-formatted evaluation from Qwen3-14B on three dimensions:

1. Fluency — grammaticality and naturalness
2. Relevance — alignment with the intent and content of the incoming email
3. Politeness — appropriateness and professionalism of tone

The scores are given on a 1-5 Likert scale.

3. Experiments

3.1. Data Augmentation

The goal of the data augmentation experiment is to evaluate whether simple word-level perturbations can improve model robustness and generalization in both spam classification and email reply generation tasks.

3.1.1. Data Augmentation Strategies

To improve model robustness under limited labeled data, we apply a simple word-level data augmentation pipeline to the email text before training. For each input message, we independently sample one of three operations:

1. **Synonym Replacement**, where up to n non-stopword tokens are replaced with WordNet synonyms
2. **Random Deletion**, where each token is removed with probability p

3. **Random Combination**, which randomly applies synonym replacement, random deletion, or both in sequence

For the spam classification task, augmentation is applied only to the Message field in the EnronSpam dataset, and the augmented messages are used to further train the DistilBERT spam classifier while keeping labels unchanged.

For the reply generation task, augmentation is applied only to the incoming email Message in the Synthetic Reply dataset, while the corresponding reply text remains fixed, so that GPT-2 learns to map a more diverse set of surface realizations of the same intent to a stable professional reply.

The augmented datasets are then integrated into the same training and evaluation pipeline used for the baseline models.

3.1.2. Evaluation Results

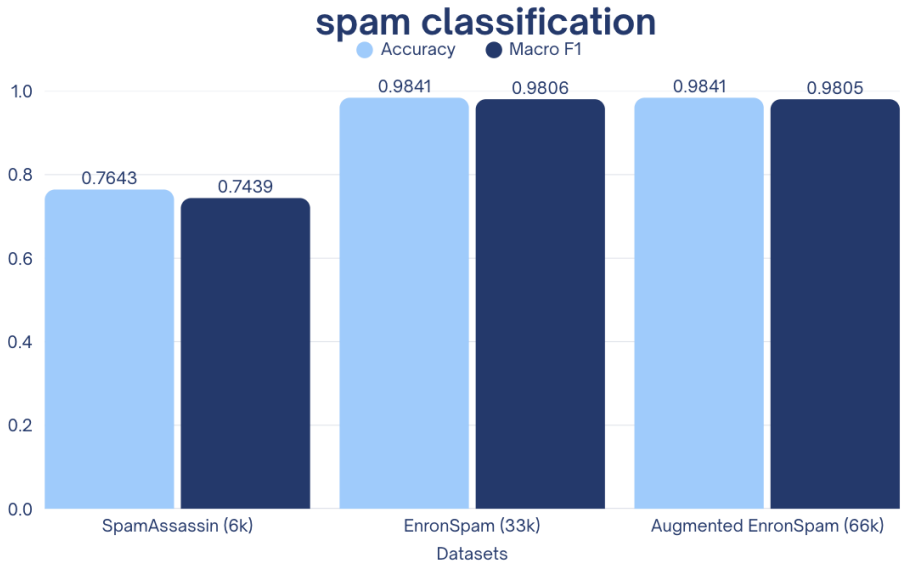

Figure 2: Evaluation results for the spam classification task with data augmentation

For the spam classification task (**Figure 2**), the model trained on the small SpamAssassin corpus (6k emails) achieves moderate performance, with about 0.76 accuracy and 0.74 macro F1.

When switching to the much larger EnronSpam dataset (33k emails), both metrics jump to around 0.98, showing that access to more diverse real emails is the dominant factor for improving the classifier. This result also indicates that the dataset is sufficiently large and diverse for training a high-quality DistilBERT classifier.

When doubling the dataset to 66k samples through simple word-level augmentation, the metrics remain essentially unchanged. The primary reason for the lack of improvement may not be the augmentation method itself, but rather that the baseline dataset is already large enough for the model to reach near-saturated performance.

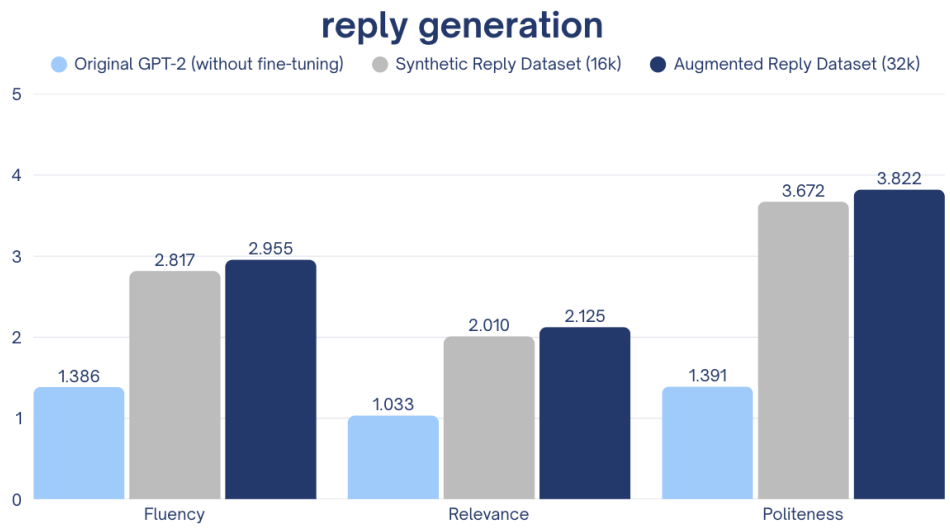

Figure 3: Evaluation results for the reply generation task with data augmentation

For the reply generation task (**Figure 3**), the effect of fine-tuning and augmentation is much more pronounced. The original GPT-2 model, without any fine-tuning, receives very low LLM-based scores (fluency ≈ 1.39, relevance ≈ 1.03, politeness ≈ 1.39 on a 1–5 Likert scale). Fine-tuning on the Synthetic Reply Dataset (16k pairs) substantially improves all three dimensions, raising fluency and relevance to around 2.8 and 2.0, and politeness to about 3.7. Further training with the augmented reply dataset (32k pairs) brings consistent but smaller gains (fluency ≈ 3.0, relevance ≈ 2.1, politeness ≈ 3.8), indicating that data augmentation helps the model generate slightly more natural, on-topic, and polite replies on top of the benefit from fine-tuning.

3.2. Zero-Shot vs. Few-Shot vs. Fine-Tuned

As the mechanism of classification model and generation model is different, here we illustrate the experiment from different model's view.

3.2.1 Experiments on Classification Model

Experiment Aim:

This experiment is conducted to study how the performance of a pretrained language model on spam detection varies under different training settings, ranging from no labeled data (zero-shot) to limited labeled examples (few-shot) and full-dataset fine-tuning. We also further develop scaling experiments to quantify the impact of training data scale on classification accuracy.

Experiment Settings:

We divided the experiment into three groups: Zero-Shot group with DistilBERT model without finetuning, Few-Shot group with DistilBERT model trains on several

numbers of training sample and Fine-Tuned group with the classification model fully finetunes on the whole dataset.

For the few-shot groups, we dive deeper into quantitative analysis. Through systematically increasing the number of training samples from 4 to 128 across multiple scales, we can see the tendency more precisely. This scaling design aims to further uncover performance patterns as the amount of labeled data grows.

All of those experiments were conducted on Kaggle with *T4 x 2 GPU, which has 15360MB x2 GPU memory.*

Evaluation Results:

As you can see from the figure2. A clear upward trend in accuracy is observed across the Zero-Shot, Few-Shot, and Fine-Tuned groups. This shows that the base DistilBERT model performs poorly on spam detection. However, as the model is fine-tuned on increasing amounts of data, the performance improves progressively, ultimately achieving remarkable accuracy when fully fine-tuned on the entire dataset.

Within the Few-Shot groups, we observe clear scaling effects as the number of training samples increases. All experiments were conducted with a fixed random seed (seed = 99) to ensure reproducibility. As shown in the graphs, model performance remains stable before 16 training samples. After 16 samples, it exhibits a consistent upward trend. This pattern consolidates our key finding: The performance of DistilBERT on spam detection is highly dependent on supervision and follows a clear data-scaling law.

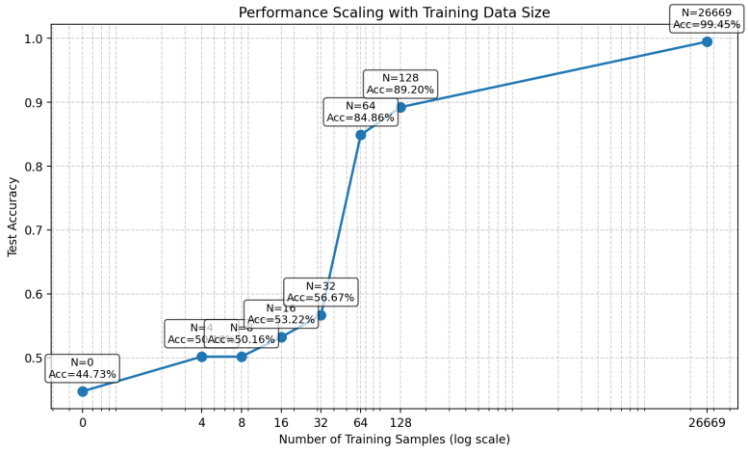

Figure 4: The relationship between model performance and training samples

3.2.1 Experiments on Generation Model:

Experiment Aim:

This experiment aims to evaluate the effectiveness of fine-tuning a generative language model on a domain-specific email replies corpus. By comparing its zero-shot generation capabilities against its fine-tuned version, we evaluate the response quality, specifically for fluency, contextual relevance, and politeness.

Experiment Settings:

Here we only conduct comparisons between zero-shot and fine-tuned generation models. The zero-shot model is the original GPT-2 model without finetuning; the fune-tuned model is the final model we used, which is trained on the whole email replies corpus. And we evaluate the performance of each model after training.

Evaluation Results:

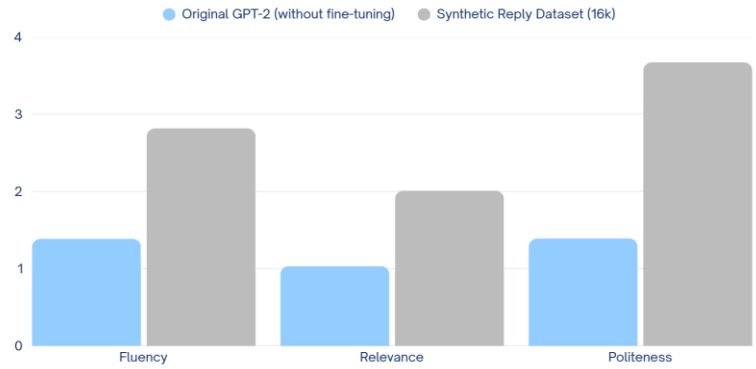

Figure 5: The performance of zero-shot model and fine-tuned model

As shown in Figure 3, the fine-tuned model exhibits a dramatic improvement across all three evaluation dimensions: fluency, relevance, and politeness.

These results highlight that while the base GPT-2 model can generate fluent text, it lacks task-specific behaviors such as relevance and politeness. Fine-tuning on a targeted dataset effectively transfers these desired traits, leading to a substantial and measurable enhancement in overall response quality.

3.3. LoRA vs. Full-Precision Fine-Tuning

Experiment Aim:

This experiment is conducted to study the memory usage and training time difference between LoRA fine-tuning and Full-Precision Fine-Tuning. We conducted experiments on both classification models and generation models. The only difference is whether to adopt LoRA or not, and the goal is to find out the effect on memory usage and training time difference, so we will discuss the experiment on classification model and generation model together.

Experiment Settings:

For classification model, we apply LoRA to all attention linear layers in the transformer blocks—namely q_lin, k_lin, v_lin, and out_lin across all 6 layers—using a rank $r = 16$, scaling factor $\alpha = 32$, and dropout rate of 0.05.

For generation model, we apply LoRA to all attention and feed-forward linear projection layers in the transformer blocks—namely c_attn (which jointly projects queries, keys, and values) and c_proj (the output projection in both the attention and

MLP submodules) across all 12 layers—using a rank r = 8, scaling factor \alpha = 16, and dropout rate of 0.05.

All of those experiments were conducted on Kaggle with T4 x 2 GPU, which has 15360MB x2 GPU memory.

Evaluation Result:

DistilBERT	GPU memory usage	CPU memory usage	Training time
Full-Precision Fine-tuning	21634 M	1505.5 M	1428 s
LoRA Fine-tuning	13766 M	868.1 M	308 s

Table 1: The resource cost of different training setting (DistilBERT)

GPT-2	GPU memory usage	CPU memory usage	Training time
Full-Precision Fine-tuning	9254 MB	690.5 MB	1815 s
LoRA Fine-tuning	6290 MB	537.3 MB	581 s

Table 2: The resource cost of different training setting (GPT-2)

For DistilBERT, the LoRA trainable parameters amount to 1,181,954 out of a total of 68,136,964, representing 1.73% of the model. In contrast, for GPT-2, LoRA introduces 811,008 trainable parameters out of 125,250,816 total parameters, which accounts for only 0.65% of the model.

For classification tasks, LoRA reduced GPU memory usage by approximately 36%, while CPU memory usage dropped by around 42%. For generation models, the GPU memory consumption decreased by about 32%. Notably, LoRA also drastically cut down training time. The training time of classification model and generation model decreased 78% and 68% respectively.

These findings highlight that LoRA is not only highly efficient in reducing resource demands but also accelerates the training process significantly, making it an ideal choice for scenarios constrained by computational resources or requiring rapid experimentation.

4.Conclusion

In this work, we developed SmartMail, an end-to-end intelligent email processing system that integrates spam detection, automatic reply generation, and a practical web-based interface. By fine-tuning DistilBERT and GPT-2 on domain-specific email corpora, we demonstrated that both models can be effectively adapted to real-world email management tasks. Our experiments show that large, high-quality datasets play a critical role in achieving strong classification performance, while reply generation benefits substantially from supervised fine-tuning and further improves with data augmentation. Additional studies on zero-shot, few-shot, and LoRA-based training highlight the importance of supervision scale and efficient parameter tuning techniques for resource-constrained environments. Overall, SmartMail presents an effective and extensible framework for intelligent email automation, and future work may explore more advanced architectures, richer human-feedback signals, and

broader multilingual or multimodal capabilities.

Appendix

1. Scoring Prompt for Reply Generation Evaluation

```
prompt = f"""
You are an expert email writing evaluator.
I will give you:
1) An incoming email.
2) A model-generated reply.
{ '3) A reference human reply.' if ref_part else '' }
Please evaluate ONLY the model-generated reply on three dimensions, using a 1–5 point Likert scale (integers only):
- Fluency: Is the reply grammatically correct and natural?
- Relevance: Does the reply correctly address the content and intent of the incoming email?
- Politeness: Is the reply polite and appropriate in tone?
Return your answer as a JSON object ONLY, with this exact format:
{{
   "fluency": <integer 1-5>,
   "relevance": <integer 1-5>,
   "politeness": <integer 1-5>
}}
Do not add any explanation or extra text outside the JSON.
Incoming email:
{incoming_email}
Model-generated reply:
{model_reply}
{ref_part}
"""
```

2. SmartMail Web Interface

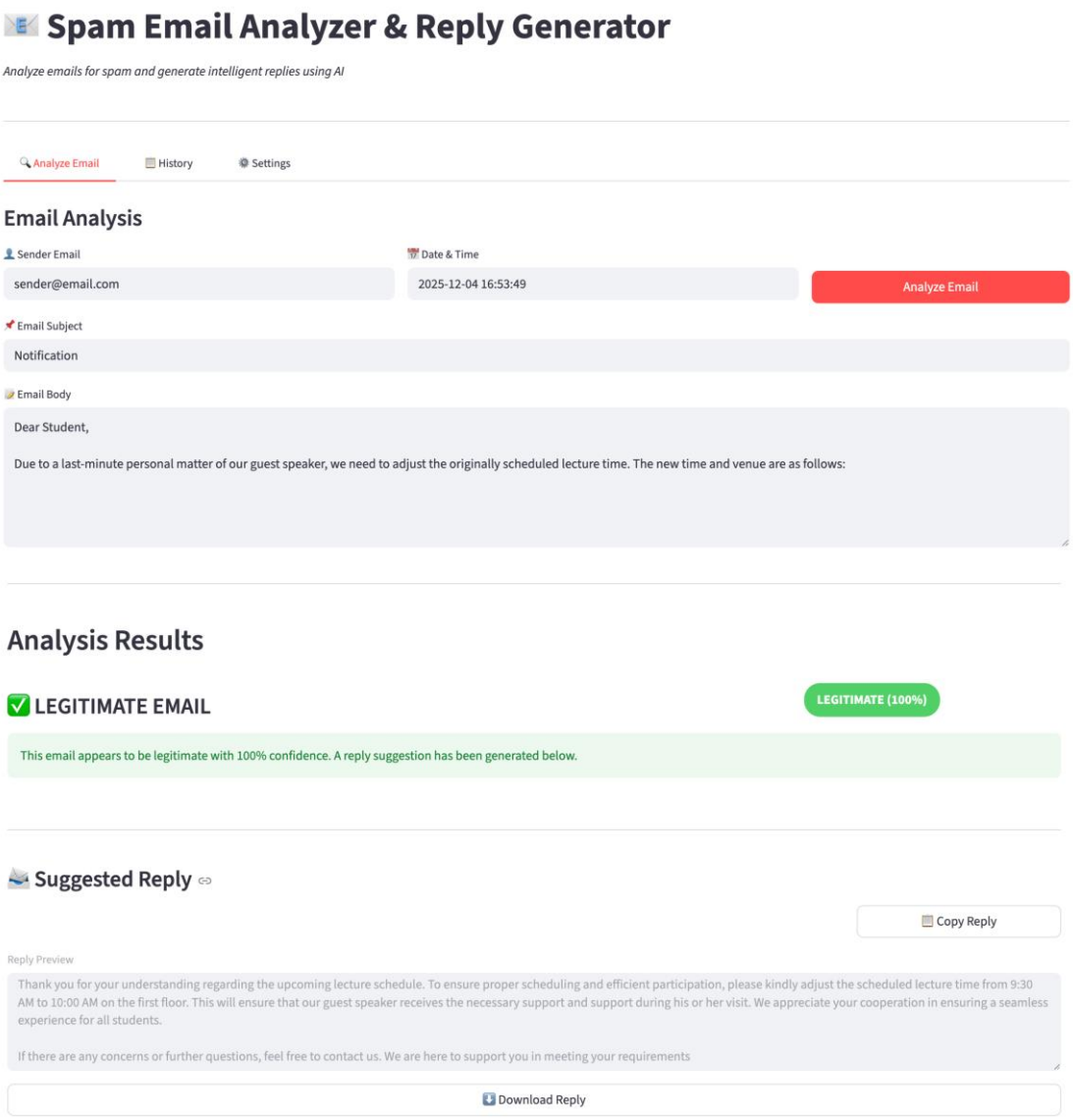

Figure 6: Web Interface for SmartMail