Northumbria University
NEWCASTLE

Local Email System with a Cascade Machine Learning Spam Classifier

Word count: 12,800

Abdulla AlBassam

Supervised by: Dr. Longzhi Yang

BSc (Honours) Networks and Cyber Security
Academic Year 2025–26
Computing Project

Acknowledgements

I would like to thank my family, friends, roommates, cat, and bunny for their patience as I attempted to wrap my head around the theory and application of machine learning. I am also grateful to my supervisor for his continued guidance and support throughout this academic year.

Abstract

Spam detection has largely been studied at the classifier level, against perturbed datasets in benchmark settings. This project built and verified a hardened local mail system with an integrated AI cascade spam filter, evaluating it as a working product rather than as a detached classifier.

A Postfix and Dovecot stack provides authenticated submission over TLS, encrypted mailbox access and a full SPF, DKIM and DMARC chain. A cascade filter pairs a TF-IDF and Logistic Regression baseline with fine-tuned DistilBERT, escalating only when baseline confidence falls below 0.90; a Postfix milter applies verdicts and a Dovecot Sieve rule quarantines spam. Verification covered four layers: transport and authentication at SMTP and IMAPS, both model pipelines through cross-validation and held-out evaluation, the adversarial generator against target perturbation rates, and cascade integration through a white-box suite.

The two models were tested against nine adversarial conditions. The baseline reached 99.09% accuracy on clean data, DistilBERT 99.55%. Character perturbation and synonym substitution had negligible effect, whereas text dilution fell to 89.23% at heavy intensity while DistilBERT retained 96.81%, a 7.58 percentage-point gap. DistilBERT's accuracy-retention advantage over a TF-IDF baseline concentrates in content-injection attacks rather than distributing uniformly across perturbation types, validating the cascade design within the project's evaluation.

End-to-end validation of the running system used a stratified sample of 900 messages submitted via authenticated SMTP and read back over IMAPS, producing a system-level accuracy of 99.6%, similar to DistilBERT's benchmark accuracy. The four false negatives concentrated in heavy and medium text dilution, matching the dataset-level finding.

Keywords: email security, spam classification, DistilBERT, TF-IDF, cascade classifier, machine learning, cybersecurity.

Table of Contents

1. Introduction

1.1 Background and Problem Context

Email remains a critical communications channel for individuals and organisations, and is correspondingly attractive for abuse. Unsolicited bulk emails and targeted phishing impose operational cost, user harm, and security risk, while defenders must balance filtering aggressiveness against the consequences of false positives (Wang, 2024). Modern filtering pipelines increasingly incorporate machine learning to augment heuristic and rule-based checks (Jáñez-Martino et al., 2023), but these classifiers operate on input that the adversary directly authors. The attacker controls the artefact under measurement and can iterate cheaply, modifying content to evade automated detection while keeping the message believable to a human recipient. Consequently, high accuracy on clean benchmark data is not sufficient to demonstrate operational resilience; accuracy retention under realistic text manipulation is a requirement for security-critical deployment (Hotoğlu et al., 2025).

1.2 Aims and Objectives

The primary aim of this project was to build a hardened, reproducible, local email system with an integrated cascade AI spam filter, running on a single host, and functionally tested end-to-end across its constituent layers. A product-led approach was taken because operational filtering depends on the surrounding environment (transport security, authentication, mail routing), not the classifier alone.

As part of evaluating the spam-filter component, the project also addressed a supporting aim: whether DistilBERT retains accuracy better than a TF-IDF and Logistic Regression baseline for email spam detection under realistic text manipulation. Accuracy retention was measured as the proportion of predictive performance held when inputs were perturbed under otherwise identical evaluation conditions. The supporting aim informed the design choice between using a single classifier and using a cascade of two, and its findings are reported

along with the product evaluation.

The objectives that follow from these aims are:

1. Build a secured local email environment (SMTP/IMAP, TLS, SPF/DKIM/DMARC).
2. Justify a labelled dataset strategy for spam/ham classification.
3. Build a reproducible preprocessing pipeline producing train/validation/test splits.
4. Implement a TF-IDF and Logistic Regression baseline and establish clean-data performance.
5. Fine-tune DistilBERT and compare against the baseline on the same sets.
6. Implement an automated adversarial attack suite (character-level, synonym, content injection).
7. Evaluate both models on clean and adversarial data using standard metrics.
8. Integrate the cascade as a Postfix milter with Sieve quarantine to Junk.
9. Evaluate functionality (latency, throughput, CPU and RAM).
10. Produce a complete evidence folder and dissertation report.

1.3 Scope and Contributions

The scope of this project was bounded by what could be built and verified within the time constraints of an undergraduate computing project. The system handled binary spam classification (spam or ham) for English-language email and was developed and deployed on a containerised system running on a personal computer. The adversarial inputs were generated programmatically rather than collected from in-the-wild attacks, the corpus used for training and evaluation was the Enron-Spam dataset, and no model-level defensive mechanisms beyond the cascade itself, such as adversarial training or ensembling, were applied. External interoperability with production mail transfer agents (MTAs) and sustained-load behaviour are therefore extrapolated rather than tested directly.

Three contributions follow. First, the product itself: a containerised local email system in which authentication, the cascade ML spam filter and Sieve quarantine routing operate end-to-end, paired with a four-layer testing approach. End-to-end validation through the live mail stack reached 99.6% system-level accuracy across 900 messages submitted via SMTP and read back via IMAPS. Second, a comparative accuracy-retention finding within the project's

adversarial suite. Third, a controlled adversarial suite of nine reproducible test conditions with fixed seed and retained generator script, applicable to future work.

2. Research and Planning

2.1 Literature Review

2.1.1 Traditional Machine Learning for Spam Detection

Machine learning has dominated spam detection for nearly three decades. Sahami et al. (1998) introduced probabilistic Bayesian classification to email filtering, and Metsis et al. (2006) benchmarked five Naive Bayes variants on the Enron-Spam corpus, reporting 97.53% average spam recall for Multinomial Naive Bayes with Boolean attributes. These early studies established the bag-of-words paradigm that still drives traditional spam classification: emails are converted into numerical feature vectors based on word frequencies or Term Frequency-Inverse Document Frequency (TF-IDF) weightings, ignoring word order and contextual relationships.

On the strength of this paradigm, traditional classifiers remain competitive on standard benchmarks. Kshirsagar et al. (2025) reported SVM with TF-IDF achieving 97.7% accuracy on Enron-Spam, with deep learning ahead by only one percentage point; ensemble methods and metaheuristic optimisation push optimised baselines beyond 98% on Enron-Spam and SpamAssassin (Manita et al., 2023; Adnan et al., 2024; Al-augby et al., 2025). The performance ceiling on clean, well-structured datasets is therefore both high and well established.

However, this consensus on clean-data accuracy obscures a fundamental architectural constraint. These models rely on surface-level statistical patterns rather than semantic understanding of email content; Tian et al. (2025) explicitly acknowledge "poor generalization capabilities" when methods are deployed in real-world conditions. If classification decisions are driven by the statistical distribution of specific tokens, then deliberate manipulation through character substitution, synonym replacement, or content injection can systematically undermine model performance. Prior work confirms that traditional classifiers are highly susceptible to such adversarial manipulation (Kuchipudi et al., 2020; Gu et al., 2021; Jáñez-Martino et al., 2023).

2.1.2 Transformer-based Approaches to Text Classification

The limitations of bag-of-words representations have motivated transformer-based architectures, introduced by Vaswani et al. (2017). Self-attention lets every token in a sequence attend to every other token simultaneously, capturing long-range dependencies that recurrent and convolutional approaches struggle to model. Devlin et al. (2019) adapted this into BERT (Bidirectional Encoder Representations from Transformers), which pre-trains deep bidirectional representations on large unlabelled corpora and is fine-tuned on downstream tasks. Rather than treating text as an unordered collection of independent term weights, BERT generates contextual embeddings in which each word's representation is informed by its surroundings.

BERT's computational requirements present practical constraints for resource-limited environments. Sanh et al. (2019) addressed this through knowledge distillation, producing DistilBERT, which retains 97% of BERT's language-understanding capability while reducing model size by 40% and increasing inference speed by 60%. Sbei et al. (2024) further found that smaller distilled models can outperform larger counterparts on standard tasks. Although alternative BERT derivatives exist, including RoBERTa and ALBERT, DistilBERT has emerged as the widely adopted variant where computational resources are a constraining factor.

Empirical evidence supports transformers for email and text classification specifically. Garrido-Merchán et al. (2023) found that BERT consistently outperformed TF-IDF with Support Vector Machines and Logistic Regression on the IMDB benchmark and three other corpora. For email security, Alhuzali et al. (2025) reported BERT (98.99%) and RoBERTa (99.08%) outperforming traditional machine learning by an average margin of 4.7% across a fourteen-model evaluation, and Jamal et al. (2024) demonstrated strong detection rates for fine-tuned DistilBERT and RoBERTa on phishing and spam emails.

However, superior clean-data performance does not necessarily translate to adversarial resilience. Jin et al. (2020) demonstrated that BERT's accuracy on sentiment classification (IMDB) dropped from 90.9% to 13.6% under synonym-based adversarial perturbations, suggesting that the contextual understanding provided by self-attention does not fully protect against targeted input perturbations.

2.1.3 Adversarial Attacks on Text-based Classifiers

Adversarial NLP attacks exploit text classifiers' sensitivity to small input perturbations that preserve human interpretability while inducing misclassification (Qiu et al., 2022; Jin et al.,

2020). Unlike continuous-domain attacks such as those on images, textual perturbations are discrete edits to tokens rather than gradient-based shifts, judged on semantic preservation and fluency (Morris et al., 2020b). The literature taxonomises attacks by semantic granularity (character, word, sentence, multi-level) and by attacker capability, ranging from black-box settings in which the model is queried as an oracle to stronger threat models with richer feedback (Qiu et al., 2022). Tooling that formalises attacks as modular pipelines with explicit goals, constraints, transformations and search strategies, such as TextAttack, have helped make these comparisons more reproducible (Morris et al., 2020a).

Concrete attack designs span the granularity spectrum. At the character level, Gao et al. (2018) introduced DeepWordBug as a black-box method that ranks tokens by importance and applies minimal edits, demonstrating that strong degradation is achievable without gradient access. Word-level attacks shift to semantic-preserving substitutions: TextFooler (Jin et al., 2020) replaces critical tokens with meaning-preserving alternatives subject to grammaticality constraints. Transformer-based attacks then use masked language models to propose context-aware substitutions: BERT-Attack (Li et al., 2020) and BAE (Garg and Ramakrishnan, 2020) report fluent perturbations against fine-tuned BERT classifiers, with smaller perturbation rates than rule-based methods. Together these results show that high clean-data performance is not a sufficient indicator of reliability under active evasion.

Realistic threat models also concern attacker observability. Many deployed systems expose only the final label (spam or ham), making query-efficient hard-label attacks particularly relevant: Qiu et al. (2025) propose QEAttack, a genetic-algorithm method operating under hard-label feedback. Spam-specific work pushes further still, beyond token-level edits to sentence rewriting and AI-generated content (Hotoğlu et al., 2025), and adaptive defensive frameworks such as SMART (Taha, 2025) treat robustness as a moving target by combining semantic enrichment with multi-objective adversarial training. Taken together, these studies indicate that adversarial robustness in text classification is best understood as the relationship between perturbation mechanism, attacker constraint, and the defensibility of model decision rules under realistic language manipulation.

2.1.4 Email Security Infrastructure

Email delivery is mediated by a distributed ecosystem of Mail Transfer Agents (MTAs) exchanging messages over the Simple Mail Transfer Protocol (SMTP). SMTP was not designed with confidentiality or robust endpoint authentication, and security properties have been retrofitted through extensions and auxiliary mechanisms (Klensin, 2008; Hoffman, 2002). Durumeric et al. (2015) characterise this as a "security patchwork" in which adoption is

voluntary and the protocol culture prioritises deliverability, producing deployments that fall short of best practice. The security of any particular message therefore depends on a chain of independently administered servers, so weaknesses in transport configuration can persist even when large providers adopt stronger defaults.

Transport-layer protection between MTAs is most commonly provided through STARTTLS, which upgrades a plaintext SMTP session to a Transport Layer Security (TLS) channel (Hoffman, 2002). STARTTLS reduces exposure to passive interception when successfully negotiated, but its negotiation begins in plaintext and can be subverted by active downgrade, for example by suppressing the "250 STARTTLS" capability advertisement to force cleartext delivery (Hoffman, 2002; Margolis et al., 2018). Empirical evidence shows that "supporting TLS" does not guarantee secure delivery: while major providers increasingly encrypt high volumes of mail, a long tail of SMTP servers exhibit poor configuration with only a minority correctly authenticated. SMTP also lacks a native mechanism to enforce strict transport security or warn senders when messages traverse insecure paths, creating practical conditions in which downgrade can occur silently (Durumeric et al., 2015).

In response to these limitations, stricter transport-security mechanisms have been proposed to provide downgrade resistance and receiver authentication. MTA-STS allows recipient domains to publish a policy via DNS and HTTPS declaring that inbound mail should be delivered using TLS with valid PKIX certificates and specifying acceptable MX hosts (Margolis et al., 2018). Ashiq et al. (2025) demonstrate that although MTA-STS is actively adopted by major providers and avoids DNSSEC dependency, its deployment is operationally complex because correct setup spans DNS, HTTPS policy hosting and MX alignment. Their longitudinal measurement across tens of millions of domains shows that 29.6% publish MTA-STS records incorrectly and 3.2% are likely to experience delivery failure with compliant senders; validation failures can prompt senders to revert to opportunistic encryption, undermining the intended strictness.

Transport security mitigates interception and downgrade but does not prevent sender spoofing. SPF, DKIM and DMARC address this by aligning DNS-published policy with observed sending behaviour (Kitterman, 2014; Crocker et al., 2011; Kucherawy and Zwicky, 2015). Durumeric et al. (2015) show that large providers can validate a high proportion of mail using SPF and DKIM, but adoption and policy strictness were historically limited in the broader ecosystem. Even where authentication is deployed, it is often weakened by permissive configuration: large-scale SPF measurement finds both syntactic and semantic misconfiguration at Internet scale, including overly permissive policies that authorise extremely large IP ranges (Czybik et al., 2023), and shared IP pools in cloud and proxy infrastructure

can magnify SPF weaknesses, enabling bypass against permissively-configured domains (Wang et al., 2024).

Overall, the literature portrays email security infrastructure as a layered set of partially independent controls spanning transport confidentiality, receiver and sender authentication, with outcomes determined by deployment accuracy and ecosystem coordination (Durumeric et al., 2015; Ashiq et al., 2025; Czybik et al., 2023; Wang et al., 2024). STARTTLS provides broad but opportunistic encryption; MTA-STS introduces enforceable policy at the cost of operational complexity (Margolis et al., 2018; Ashiq et al., 2025); SPF, DKIM and DMARC reduce spoofing risk but remain sensitive to misconfiguration and shared infrastructure effects (Czybik et al., 2023; Wang et al., 2024). These deployment realities motivate this project's choice to embed the spam filter within a hardened, locally controlled mail stack rather than treat classification as a detached benchmark task.

2.1.5 Summary and Project Rationale

Across these themes, the literature shows that clean-data accuracy is no longer the central question in spam detection. Traditional classifiers do well on benchmark text but rely on lexical cues that adversarial manipulation easily disrupts (Kuchipudi et al., 2020; Jáñez-Martino et al., 2023). Transformer models are more robust on standard tasks but can still fail under targeted perturbation (Jin et al., 2020). Mail itself sits in a layered system in which transport security, authentication and deployment practice shape outcomes (Durumeric et al., 2015). These practical considerations, together with the model-comparison and adversarial-attack literature reviewed above, motivated this project's choice to evaluate classical and transformer classifiers as components of an end-to-end mail system under a security-realistic threat model rather than as detached benchmarks.

2.1.6 Tools and Technologies

The practical work used a containerised stack so the system is reproducible from versioned infrastructure-as-code. Docker Desktop and Docker Compose deploy the local mail system; docker-mailserver provides the integrated Postfix and Dovecot deployment with OpenDKIM, OpenDMARC, policyd-spf and Fail2ban. Server-side filtering uses Dovecot Pigeonhole Sieve. The supporting containers are Roundcube for webmail and dnsmasq for the local DNS zone.

Model development used Python: scikit-learn (*TfidfVectorizer*, *LogisticRegression*) for the baseline, HuggingFace Transformers and PyTorch for DistilBERT fine-tuning on a Google Colab T4 GPU. The adversarial generator was a custom Python script using NLTK/WordNet

(synonyms), the standard library (character perturbation) and a fixed lexicon (text dilution); TextAttack (Morris et al., 2020a) was rejected as model-specific, with the full rationale in Section 3.1.4. For milter integration, pymilter was selected so the cascade classifiers run in-process and avoid an IPC boundary with libmilter. Data handling, plotting and model persistence use pandas, numpy, matplotlib, seaborn and joblib.

2.2 Requirements

The functional requirements describe the externally observable behaviour of the deployed mail stack and cascade filter; the non-functional requirements describe constraints on how that behaviour is delivered and verified. Both sets are referred back to in Chapters 3 and 4 when each layer is implemented and tested.

2.2.1 Functional Requirements

1. The system shall accept authenticated message submission over TLS-encrypted SMTP, with SMTP AUTH not advertised before STARTTLS and unauthenticated relay refused.
2. The system shall provide encrypted mailbox access over IMAPS and shall refuse plaintext IMAP.
3. The system shall sign outbound mail with DKIM and shall evaluate inbound SPF and DMARC during the SMTP transaction, applying the DMARC reject policy on alignment failure.
4. The system shall classify each inbound message as spam or ham using a two-stage cascade: a TF-IDF and Logistic Regression baseline, escalating to fine-tuned DistilBERT.
5. The system shall annotate every classified message with *X-Spam-Status*, *X-Spam-Score* and *X-Spam-Model* headers reporting the verdict, score, and the model that produced it.
6. The system shall route messages tagged *X-Spam-Status: Yes* to the user's Junk folder via a server-side Pigeonhole Sieve rule, with ham delivered to Inbox.
7. The system shall fail open on classifier failure.
8. The system shall expose a reproducible adversarial generator producing nine fixed test conditions from a seeded random state, with the generator script retained in the project repository.

2.2.2 Non-Functional Requirements

1. Reproducibility: container tags pinned to specific versions, random seeds fixed across data preparation and model training, and identical model artefacts at training and deployment time so verified and deployed models are guaranteed identical.
2. Verifiability: each product layer (mail-security stack, ML pipelines, adversarial generator, milter integration) is testable in isolation as well as end-to-end, supporting the four-layer testing approach in Section 3.3.
3. Modularity: the classifier, the milter container, and the Sieve routing rule are independently replaceable, so that future work does not require restructuring the surrounding system.
4. Security: transport security shall enforce TLS 1.2 or higher with TLS 1.0 and 1.1 refused at the protocol layer; plaintext authentication shall be disabled across SMTP submission and IMAP; sender authentication shall be enabled through SPF, DKIM signing and DMARC with a reject policy; and the spam-classification path shall fail open on classifier failure (deliver mail), while the transport layer shall fail closed on missing or weak TLS so that delivery cannot silently downgrade.

3. Practical Work

3.1 Design

The system was built around three artefacts: a hardened mail server, a cascade spam filter that runs in the message-acceptance path, and an adversarial test-set generator.

3.1.1 System Architecture

The system is presented at two levels of detail. Figure 3.1 shows the conceptual data flow from sender to mailbox where a message enters via authenticated SMTP, passes through the cascade spam classifier, receives spam-status headers, and is routed by a Sieve rule into Inbox or Junk. Figure 3.2 maps that flow onto the four Docker services in greater detail.

Firstly, the system fails open on classifier failure: if the milter is unreachable, times out, or raises an exception, Postfix delivers the message. Detection degradation must not become a denial-of-service vector; the worst-case effect of a classifier failure is delivery to Inbox without spam headers, not a lost message. Conversely, the system fails closed on transport-security failure: TLS 1.0 and 1.1 are refused at the protocol layer, plaintext authentication is disabled, and SMTP AUTH is not advertised before STARTTLS, so a misconfigured client cannot silently downgrade. Finally, the system is engineered for reproducibility: pinned container tags, fixed random seeds, and identical model artefacts at training and deployment time guarantee the verified and deployed models are the same.

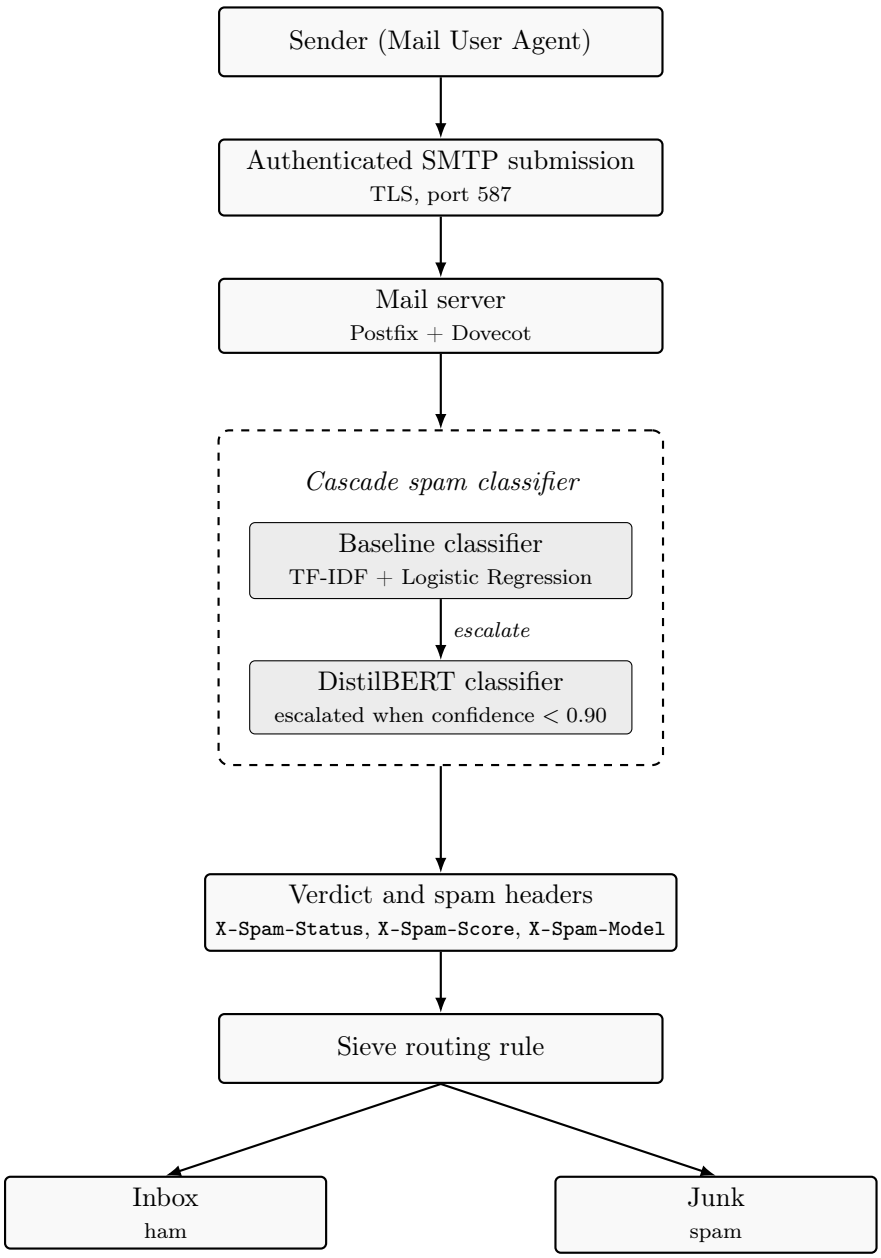


Figure 3.1: High-level block diagram of the system.

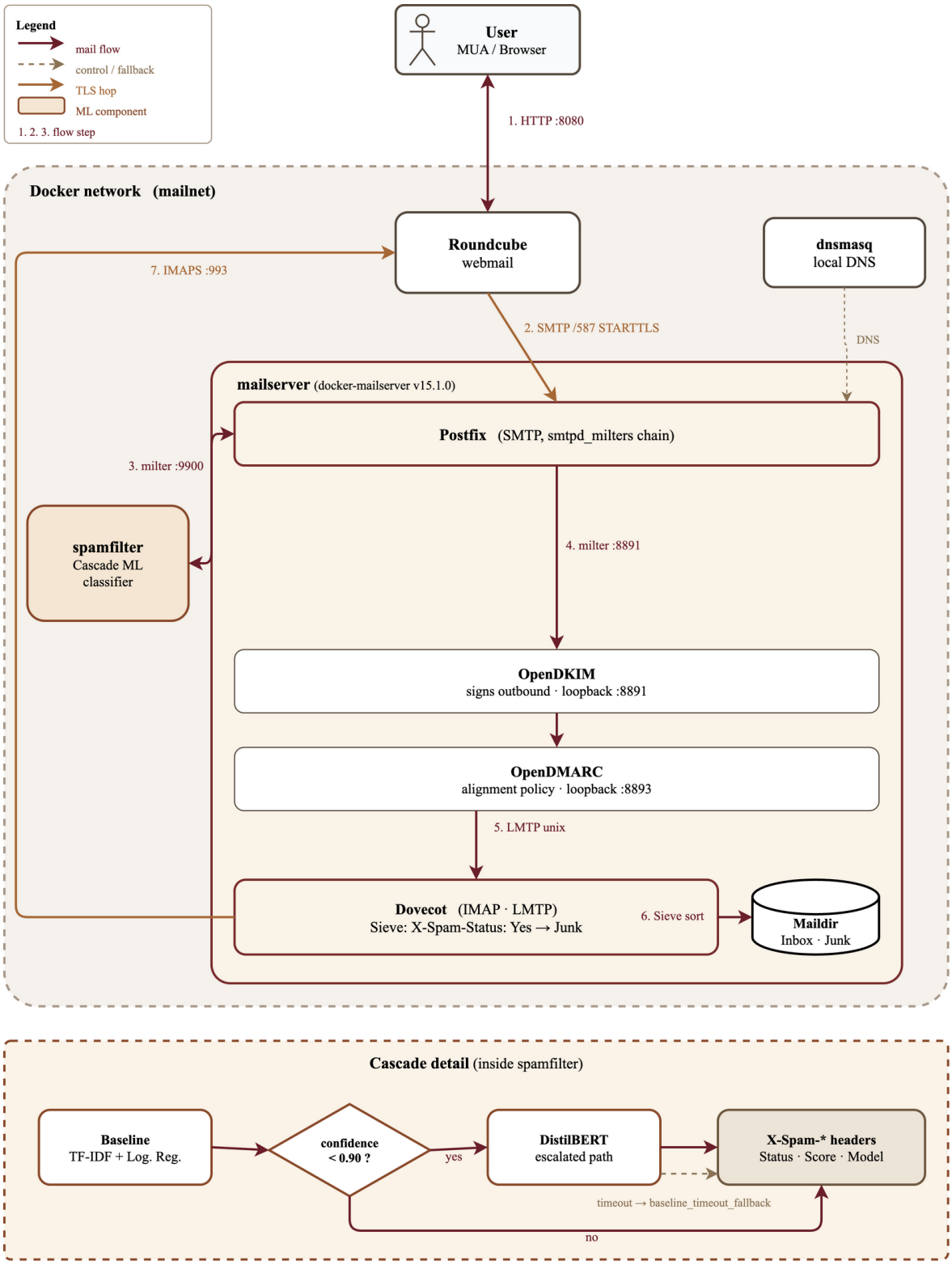


Figure 3.2: End-to-end architecture of the mail system, with the four Docker services, the milter chain inside Postfix, the Sieve routing rule, and the cascade detail.

3.1.2 Mail Server

The mail server runs on a single host as a small set of Docker services on an isolated bridge network. docker-mailserver was chosen as the Postfix and Dovecot deployment because it provides sensible defaults that can be overridden through mounted configuration. Heavier turnkey alternatives such as Mailcow and Mailu were rejected as expanding surface area beyond what the project required.

The bundled SpamAssassin, ClamAV, Postgrey and Amavis components are disabled so that the cascade milter is the sole spam-decision maker, and any change in mailbox routing can be attributed to the cascade rather than to a built-in filter. Postfix is restricted to IPv4 so that test traffic uses one address family, eliminating inconsistent results from IPv6 fallbacks. Three persistent volumes hold mailboxes, runtime state and logs across container restarts, so account state and DKIM key material survive iterative rebuilds.

The TLS profile shipped by docker-mailserver is retained: TLS 1.0 and 1.1 are refused (Moriarty and Farrell, 2021), ciphers are restricted to a high-grade set, and compression and renegotiation are disabled in line with current TLS guidance (Sheffer et al., 2022). Opportunistic TLS is kept on TCP/25 for general SMTP compatibility, but the submission service on TCP/587 enforces encryption before authentication and scopes SMTP AUTH to that service via the master.cf shipped by docker-mailserver (Postfix Project, n.d.b; Postfix Project, n.d.e), as required for submission and access (Moore and Newman, 2018). Dovecot is configured to require TLS for IMAP and to disable plaintext authentication, so credentials cannot be accepted over an unencrypted channel. Fail2ban mitigates repeated authentication failures during testing.

Email authentication is configured to produce realistic anti-spoofing signals before the spam filter runs. SPF is published as a DNS record and evaluated by OpenDMARC during DMARC alignment alongside DKIM, producing a single anti-spoofing milter outcome rather than running SPF as a separate transaction-time policy service. DKIM signing and DMARC evaluation use OpenDKIM and OpenDMARC via the standard milter sockets retained from docker-mailserver. The DKIM key is 2048-bit; a 4096-bit key would add cost without a meaningful security gain at this scale. DNS records for the test zone are served locally by a dnsmasq container so the environment is independent of external DNS. The zone hard-fails (*-all*) any sender outside the authorised set, the DKIM public key is published under the standard *_domainkey* subdomain, and the DMARC policy uses a reject action with strict alignment for both DKIM and SPF and the same reject policy for subdomains.

The Roundcube webmail client connects to the mail server using IMAPS and authenticated

submission over STARTTLS. Certificate validation is enforced rather than bypassed, the trusted CA is mounted into the Roundcube container, and both peer verification and hostname verification are enabled, preventing insecure shortcuts such as "allow self-signed."

3.1.3 Cascade Spam Filter

The spam filter is a two-stage cascade integrated as a Postfix milter. A TF-IDF and Logistic Regression baseline classifies every inbound message; a fine-tuned DistilBERT model is invoked only when baseline confidence falls below 0.90. The cascade design separates fast, transparent classification of obvious cases from a heavier transformer pass that contributes most where the baseline is least confident, so the average per-message cost is dominated by the cheap path while harder cases still benefit from contextual evaluation.

Both classifiers were trained on the Enron-Spam dataset of 33,716 labelled emails. Enron-Spam was chosen because it provides real email content, supports comparison with prior benchmarks (Metsis et al., 2006; Al-augby et al., 2025), and is roughly balanced (50.9% spam, 49.1% ham) so no resampling was needed and standard metrics could be interpreted without imbalance distortion. Subject and body were combined into one text field rather than discarding empty-body rows, since the empty-body rows were dominated by spam where persuasive content sits in the subject line (Bhowmick and Hazarika, 2018); the breakdown is in Section 3.2.4. Stemming and lemmatisation were not applied because the baseline uses TF-IDF with n-grams rather than aggressive morphological reduction; stopword removal was deferred to the vectoriser to keep it configurable. An 80/20 stratified split with a fixed random seed produced 26,929 training and 6,733 test rows and the same split was reused for both classifiers so any difference in performance can be attributed to model architecture rather than to data variation.

The baseline pairs TF-IDF with Logistic Regression. The pairing was chosen for documented effectiveness in email spam classification (Manita et al., 2023; Kshirsagar et al., 2025) and for transparency because Logistic Regression coefficients are directly interpretable as feature-level contributions to the classification decision, which matters for later analysis of how adversarial perturbations interact with surface-level token weightings. A 10,000-feature vocabulary with unigrams and bigrams captures discriminative multi-word patterns, while sublinear TF scaling and conservative document-frequency thresholds filter rare and overly common terms. L2 regularisation at the default strength is appropriate given the high feature dimensionality, penalising large coefficients and reducing the risk of overfitting to individual tokens.

DistilBERT was selected as the transformer because it retains approximately 97% of

BERT's language-understanding performance while being 60% faster and 40% smaller (Sanh et al., 2019), keeping fine-tuning practical on a free-tier Google Colab T4 GPU and inference practical in-line in the milter on a single host. The uncased variant is the right choice because the preprocessing pipeline already lowercases all text. WordPiece tokenisation matters specifically for adversarial behaviour: character-level perturbations that produce misspelled or novel words are decomposed into known subword pieces rather than mapped to a single unknown token, so the model retains a meaningful representation of the input even when the surface form is corrupted. A maximum sequence length of 256 tokens covered 73.8% of training emails in full; the remaining 26.2% are truncated, retaining the subject line and opening content which typically carry the strongest classification signal. Training followed the standard BERT fine-tuning recipe (Devlin et al., 2019). Best-F1 checkpoint selection used the held-out test set as both in-training monitor and final evaluation surface, with limited effect at near-ceiling F1 but acknowledged as a methodological limitation.

The cascade is deployed as a Postfix milter in a separate container. pymilter was chosen because the classifiers are written in Python; in-process inference avoids the IPC boundary that libmilter would otherwise introduce. The spam milter is registered ahead of the DKIM and DMARC milters so that classification operates on the message as submitted, before authentication signals are written into the headers; otherwise local SPF, DKIM and DMARC outcomes would leak into a path the model was not trained on. The milter is excluded from the non-SMTPD path so classification runs only on inbound SMTP, not on locally-generated messages.

Both classifiers are loaded eagerly at container startup. Lazy-loading DistilBERT would shift the cold-start latency of model deserialisation onto the first message that the cascade routed to it, which could exceed the milter's content timeout for that message. Eager loading also makes a controlled warmup pass possible, which matters for the threading-deadlock fix (Section 3.2.8). The 0.90 cascade threshold is a reasoned starting point given the baseline's clean accuracy: it routes the small uncertainty band around the baseline's decision boundary to DistilBERT while letting the baseline alone handle the bulk of obvious cases. Empirical tuning of the threshold against a different ham/spam mix is left to future work (Section 5.1).

The verdict is applied through three custom headers (*X-Spam-Status*, *X-Spam-Score*, *X-Spam-Model*) added through the milter's header-only modification capability. A Dovecot Pigeonhole Sieve script files messages tagged *X-Spam-Status: Yes* into Junk. Server-side Sieve was chosen over client-side filtering because it applies uniformly across IMAP clients and delivery paths (Showalter, 2008), so the spam decision is enforced for any MUA the user happens to use.

3.1.4 Adversarial Suite

The adversarial suite was designed to put both classifiers under identical pressure so any difference in accuracy retention can be attributed to model architecture rather than to variation in the inputs. The relevant adversary controls inbound message subject and body but cannot query the classifier as an oracle, cannot intercept transport, and cannot modify the deployed models or their training data; the threat is therefore the cheap, scriptable text manipulation of spam content the attacker authors. Model-specific algorithms such as TextFooler (Jin et al., 2020) and BERT-Attack (Li et al., 2020) were rejected because they pre-suppose query access and conflate classifier architecture with attack design. All attacks in the suite are model-agnostic and apply fixed rules and random selection without consulting either classifier, so both models see exactly the same nine adversarial test sets.

The suite covers three attack types at three intensity levels each: character-level perturbation (typographical evasion), synonym substitution (semantic-preserving rewriting) and text dilution (content injection). The three types span the spam-relevant manipulation surface from surface-form noise through lexical rewriting to content-shift, mapping onto distinct model behaviours: character noise probes how each tokeniser handles malformed input, synonym substitution probes contextual versus lexical decision rules, and text dilution probes how each model weighs spam-indicative tokens against benign context. Attacks are applied only to the 3,424 spam emails in the held-out test partition; the 3,309 ham emails remain unchanged because the operational scenario is an attacker modifying spam to evade detection, not manipulating legitimate mail.

Character-level target perturbation rates of 5%, 10% and 20% follow Sajad (2025), who reported a clear DistilBERT degradation curve at these tiers. Synonym substitution target rates of 1%, 3% and 5% follow Hotoğlu et al. (2025). Text dilution injects 3, 7 and 12 words from a fixed list of professional and business vocabulary at random positions in the spam body. The character attack preserves the first and last characters of each word for readability; the synonym attack restricts to single-word WordNet synonyms of matching part of speech and excludes stopwords and proper nouns, preserving word count and avoiding positional shift; the dilution attack adds new tokens rather than replacing existing ones (similar in spirit to Kuchipudi et al., 2020), preserving the original spam vocabulary while diluting spam-indicative concentration. Fixed random seeds make the suite reproducible across runs given a fixed *PYTHONHASHSEED*, and the generation script is retained in the project repository for re-execution.

3.2 Implementation

This section walks through the implementation in the order it was built: the mail-security stack (Sections 3.2.1 to 3.2.3), the cascade spam filter from offline training to in-line milter (Sections 3.2.4 to 3.2.8), and the adversarial generator (Section 3.2.9). The development host was macOS (M2, 16GB RAM) running Docker Desktop 29.1.2. Docker Compose orchestrates four containers on an isolated bridge network (172.20.0.0/24): a local DNS resolver (dnsmasq), a mailserver based on docker-mailserver, a Roundcube webmail client, and the cascade spam-filter milter. The host exposes TCP/25, 587, 993 and 8080; *PERMIT_DOCKER=network* adds the bridge to Postfix's *mynetworks*, an acceptable trade-off on a single isolated host but too permissive in any deployment with untrusted neighbours. Test mailbox accounts in the mail.local domain were created using docker-mailserver's management utility.

3.2.1 Postfix and Dovecot Configuration

Transport security uses a privately issued, manually mounted TLS certificate; runtime evidence is in Section 3.3.1. The submission service on TCP/587 enforces encryption before authentication and scopes SMTP AUTH to that service via the stock master.cf shipped by docker-mailserver. On the Dovecot side, mounted overrides set *ssl = required* and *disable_plaintext_auth = yes*.

Configuration files for the mail server are in *docker-compose.yml* and *docker/mailserver/config/*.

3.2.2 Email Authentication

SPF is published as a TXT record in the local DNS zone and evaluated by OpenDMARC during the DMARC alignment check rather than as a separate transaction-time policy service. DKIM signing and DMARC evaluation are integrated using OpenDKIM and OpenDMARC via the standard milter sockets retained from docker-mailserver, with Postfix configured to fail open if a milter becomes unreachable (*milter_default_action = accept*).

The corresponding DNS records are implemented locally using dnsmasq. The local zone publishes an SPF policy authorising the domain's MX host and the mailserver's bridge address (172.20.0.10) and hard-failing (*-all*) all other senders. A DKIM public key record is published under *mail._domainkey.mail.local*, and a DMARC policy for *_dmarc.mail.local* is configured to reject messages that fail alignment, with strict alignment for both DKIM and SPF (*adkim=s; aspf=s*), the same reject policy for subdomains (*sp=reject*), and reporting addresses set to the local postmaster mailbox. DKIM key material was generated using docker-mailserver's DKIM setup tooling.

DNS zone, OpenDKIM key tables, and Postfix milter and policy-service overrides are in *docker/dns/dnsmasq.conf*, *docker/mailserver/config/opendkim/*, and *docker/mailserver/config/user-patches.sh* respectively.

3.2.3 Web Client Integration

Roundcube was deployed as a dedicated container and connected to the mailserver over the internal Docker network. IMAP connectivity was configured explicitly as IMAPS (*ssl://mailserver:993*), and outbound mail was configured to use authenticated submission over STARTTLS on TCP/587 (*tls://mailserver:587*) with credentials derived from the active Roundcube session *(%u and %p)*. The Roundcube container's default environment variables would have submitted via plaintext SMTP on TCP/25; the mounted custom configuration overrode this, routing submission through STARTTLS on TCP/587 and enabling strict certificate verification.

The certificate authority certificate used to sign the server certificate was mounted into the Roundcube container and configured as the trusted CA file. Both peer verification and hostname verification were enabled for IMAP and SMTP connections.

The Roundcube override is in *docker/roundcube/config.inc.php*.

3.2.4 Dataset Preparation

The training corpus was the Enron-Spam dataset of 33,716 labelled emails (50.9% spam, 49.1% ham). The raw data had 289 missing subjects and 371 missing bodies; 320 of the 371 empty-body rows still had a subject, of which 319 belonged to the spam class.

Word-frequency analysis after cleaning showed clear vocabulary separation between the two classes, with ham dominated by Enron-specific business terms and spam by financial and commercial vocabulary (Figure 3.3).

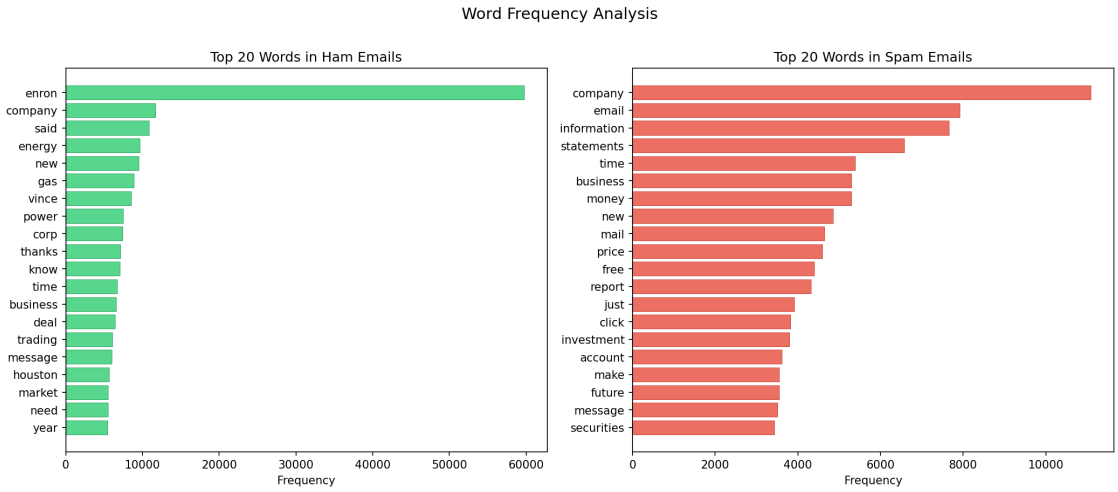


Figure 3.3: Top word frequencies by class after preprocessing, showing distinct vocabulary separation between ham (Enron-specific business terms) and spam (financial and commercial terms).

Preprocessing was implemented in a Python module reused across all notebooks and the prediction interface, applying a fixed sequence: nulls converted to empty strings, lowercasing, regex removal of email headers, addresses, URLs and HTML tags, non-alphabetic characters replaced with spaces, whitespace collapsed.

Subject and body were cleaned separately and concatenated subject-first, labels were encoded as 1 for spam and 0 for ham, and rows whose combined text was empty after cleaning were removed (54 rows, leaving 33,662). Mean text length fell from 1,470.4 to 1,316.8 characters (10.4% reduction); the median fell from 675 to 596 (Figure 3.4), indicating structural noise was removed without collapsing lexical content.

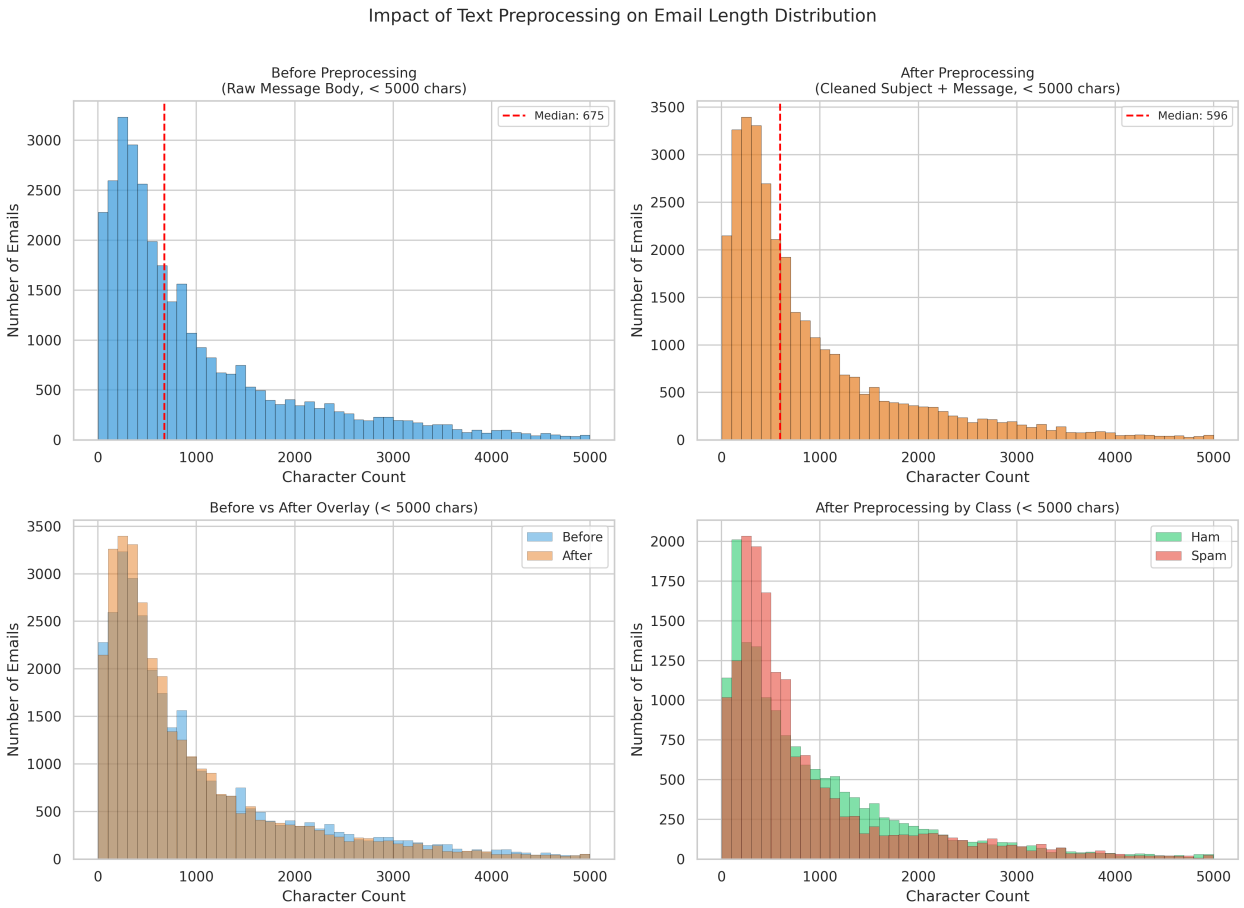


Figure 3.4: Effect of text preprocessing on email length distribution

The final prepared dataset was split into training and test partitions using an 80/20 stratified split with *random_state=42*, producing 26,929 training instances and 6,733 test instances. The training partition contained 13,236 ham and 13,693 spam emails, and the test partition contained 3,309 ham and 3,424 spam emails.

3.2.5 TF-IDF + Logistic Regression Baseline

The baseline classifier was implemented in Python using scikit-learn. Feature extraction used scikit-learn's TfidfVectorizer with a 10,000-feature vocabulary, unigrams and bigrams, sublinear TF scaling, *min_df = 2*, *max_df = 0.95*, and English stopword removal. The fitted feature matrix had 99.23% sparsity, consistent with TF-IDF over a corpus of this size.

The Logistic Regression model was trained with L2 regularisation at the default strength ($C = 1.0$), using the LBFGS solver and a maximum iteration limit of 1,000. The model

converged in 14 iterations, well within the specified limit, indicating that the optimisation landscape was well-conditioned for this dataset and feature representation.

To validate model stability before final evaluation, five-fold stratified cross-validation was conducted on the training partition only. Validation accuracy averaged 0.987 across folds (training 0.991) and F1 averaged 0.987 (training 0.991), confirming a narrow train/validation gap and no signs of overfitting.

Inspection of the model's learned coefficients, shown in Figure 3.5, revealed which TF-IDF features contributed most strongly to each classification outcome. The highest positive coefficients (indicating spam) corresponded to terms such as "http", "software", "money", "remove", and "click", while the most negative coefficients (indicating ham) corresponded to Enron-specific business terms such as "enron", "attached", "vince", "gas", and "houston". Several bigram features also appeared, including "phone mobile", "mobile email", "enron com", and "let know", confirming that the inclusion of bigrams captured multi-word patterns relevant to classification. The trained model and fitted TF-IDF vectoriser were serialised using joblib for reuse in the adversarial evaluation phase.

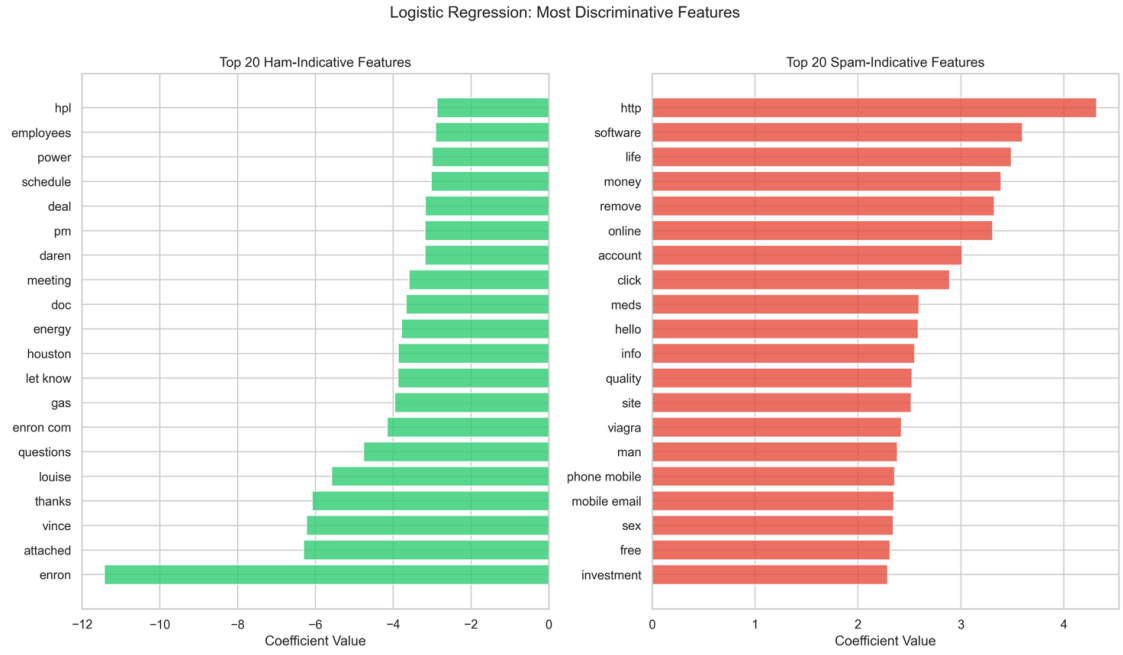


Figure 3.5: Top Logistic Regression coefficients by direction (ham vs spam) for the TF-IDF features

3.2.6 DistilBERT Fine-Tuning

DistilBERT was implemented by fine-tuning the pre-trained *distilbert-base-uncased* model for binary spam classification, using the HuggingFace Transformers library and PyTorch on a free-tier Google Colab environment with a T4 GPU (15 GB VRAM).

The DistilBERT pipeline received the same preprocessed text, the same binary labels, and the same 80/20 stratified train/test split used for the TF-IDF + Logistic Regression pipeline. The training partition contained 26,929 samples and the test partition contained 6,733 samples, with class proportions preserved. The preprocessing functions were copied directly from the shared project module into the Colab notebook to guarantee identical cleaning behaviour despite the different execution environment. Assertions in the notebook verified that the row count, split sizes, and class distributions matched the baseline exactly, guarding against environment drift between local Python and Colab that could otherwise have introduced unnoticed input differences.

Tokenisation used the DistilBERT WordPiece tokeniser, which splits input text into subword units drawn from a fixed vocabulary of 30,522 tokens. For example, the word "unsubscribe" is tokenised as *["un", "##su", "##bs", "##cr", "##ibe"]*. A maximum sequence length of 256 tokens was applied, with padding for shorter sequences and truncation for longer ones; analysis of a sample of 1,000 training texts showed that 73.8% of emails fell within 256 tokens and were therefore represented in full, with the remaining 26.2% truncated.

The tokenised data was wrapped in a custom PyTorch Dataset class returning input identifiers, attention masks, and integer labels, with the attention mask preventing padded positions from contributing to self-attention.

A two-class sequence classification head was added on top of the pre-trained DistilBERT encoder, with all weights left trainable, giving roughly 67M parameters in the fine-tuned model. Training used learning rate 2e-5, batch size 16, three epochs, weight decay 0.01, and linear warmup over the first 10% of steps; FP16 mixed precision was enabled on the T4. Evaluation ran at the end of each epoch using accuracy, precision, recall and F1 via HuggingFace's Trainer, with the best-F1 checkpoint selected automatically.

Training completed in approximately 20 minutes. The training loss fell across all three epochs (Figure 3.6), with the steepest reduction during the first epoch as the classification head adapted; no instability or divergence was observed.

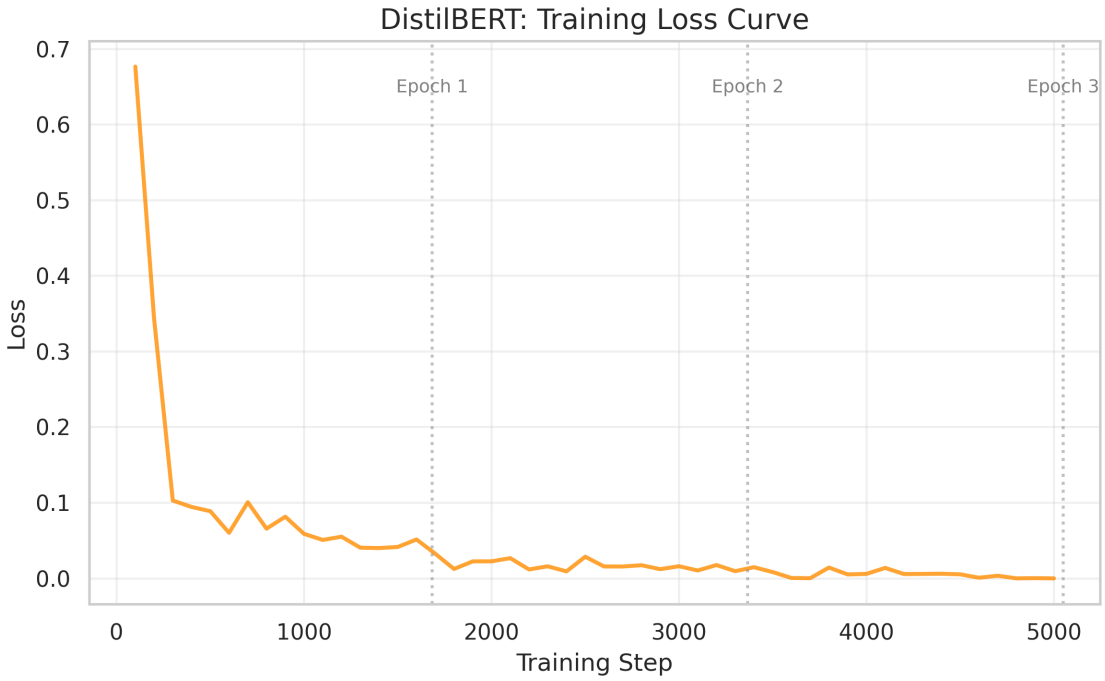


Figure 3.6: DistilBERT training loss over three epochs

Final evaluation was conducted on the same held-out test set of 6,733 emails used for the baseline. Full test set metrics are reported in Section 4.2 alongside the baseline results for direct comparison. The trained DistilBERT model and tokeniser were saved using HuggingFace's *save_pretrained* format for reuse in the adversarial evaluation phase.

3.2.7 Cascade Logic and Milter

The cascade is implemented as a separate *spamfilter* container that exposes the libmilter protocol on TCP port 9900. The milter is registered in *smtpd_milters* ahead of the DKIM and DMARC milters (*docker/mailserver/config/user-patches.sh:11*), and excluded from *non_smtpd_milters* (line 12). The container build is at *milter/*.

Both classifiers are loaded at container startup by *load_models()* in *milter/spam_milter.py*, even though the baseline alone handles the majority of emails. The eager-load also enables the warmup pass discussed in Section 3.2.8.

On *eom()*, the milter extracts *text/plain* (falling back to *text/html*, then raw) and runs the baseline; confidence *max(p, 1 - p)* below *CASCADE_THRESHOLD = 0.90* escalates to DistilBERT.

The fail-open posture is preserved at three layers. In the Postfix configuration, *mil-*

ter_ default_ action = accept (*user-patches.sh:15*) releases a message if the milter is unreachable or exceeds its connect, command, or content timeouts (30s, 30s, 180s; *user-patches.sh:19–21*). Inside the milter, *eom()* returns *Milter.ACCEPT* on any classification exception (*spam_ milter.py:265–269*), and *load_ models()* catches a DistilBERT load failure and continues in baseline-only mode (*spam_ milter.py:97*) rather than refusing to start.

The milter writes *X-Spam-Status*, *X-Spam-Score* and *X-Spam-Model* headers and is restricted to *ADDHDRS*. A Dovecot Pigeonhole Sieve script (*before.dovecot.sieve*) routes *X-Spam-Status: Yes* to Junk via *fileinto :create*.

3.2.8 Threading Deadlock

On first integration, a borderline message logged *Escalating to DistilBERT* and produced no further output: the SMTP client timed out after 30 seconds and Postfix released the message under *milter_ default_ action = accept* with no spam headers, while obvious ham and spam classified normally within 50 milliseconds. A live *py-spy* dump showed a libmilter worker suspended inside *BatchEncoding.to(self.device)*.

Three factors were at play here. libmilter (Postfix Project, n.d.f) dispatches callbacks on per-connection worker threads; PyTorch's libgomp initialises its OpenMP thread pool lazily on the first parallel operation (Free Software Foundation, n.d.); and the DistilBERT model was constructed on the main thread but never run there, so the first forward pass landed on a worker. On the Apple Silicon development host, the x86_64 PyTorch image ran under Rosetta, and libgomp's first-init path on a non-main pthread deadlocked.

The fix was three-layered. A single-worker *ThreadPoolExecutor* serialises every DistilBERT inference onto one dedicated thread, paired with a warmup pass at the end of *load_ models()* that triggers libgomp, MKL, and tokeniser lazy initialisation while the milter is idle. *OMP_ NUM_ THREADS*, *MKL_ NUM_ THREADS*, *OPENBLAS_ NUM_ THREADS* and *TOKENIZERS_ PARALLELISM* are pinned single-threaded at the container level, because libgomp reads *OMP_ NUM_ THREADS* only at *import torch* time. A 10-second executor timeout converts any residual hang into a graceful baseline fallback rather than a 180-second Postfix stall. The full forensic narrative and *py-spy* traces are documented in the project source tree under *milter/diagnostics/threading_ deadlock.md*.

3.2.9 Adversarial Suite

The adversarial suite comprised three attack types, each applied at three intensity levels, producing nine adversarial test sets. All attacks were applied exclusively to the 3,424 spam

emails in the held-out test partition; the 3,309 ham emails remained unchanged. Each adversarial dataset retained the same structure as the original test set (6,733 rows), with an additional column preserving the original unmodified text to support later analysis. All random operations were seeded deterministically; per-attack seeds combine the global seed with Python's salted *hash()*, so a fixed *PYTHONHASHSEED* is required for bit-identical reproduction across processes. The generation script is retained in the project repository for re-execution.

The first attack type was character-level perturbation. For each selected word, one of four operations was applied at random: character swap (swapping two adjacent characters), character insertion (adding a random letter), character deletion, or character substitution (replacing a character with a random alternative). To preserve human readability, the first and last characters of each word were excluded from modification, and edits were distributed across the email rather than clustered within a small number of words. The achieved perturbation rates, measured as the proportion of characters differing between original and modified text, were 8.27%, 16.41%, and 31.46% for light, medium and heavy intensities respectively. The consistent overshoot of approximately 1.6 times the target occurred because the attack perturbs one character per selected word, and short words (three to four characters) contribute a higher per-character perturbation rate than longer words. Hotoğlu et al. (2025) reported that the effect of character-level attacks on classifier accuracy plateaued beyond 30%, which supports the decision not to test above the 20% target tier. Across all three intensities, at least 99.85% of spam emails were successfully modified (3,419 of 3,424 at light, 3,422 at medium and heavy); the small number of unmodified emails contained only very short words (two characters or fewer) that were ineligible for perturbation.

The second attack type was synonym substitution. For each spam email, the attack identified eligible words using NLTK part-of-speech tagging and selected only content words (nouns, verbs, adjectives, and adverbs) of four or more characters, excluding stopwords and proper nouns. A target proportion of eligible words was then randomly selected for replacement, with WordNet queried for single-word synonyms of the same part of speech; multi-word synonyms were filtered out by excluding any WordNet lemma containing underscores, ensuring that each replacement preserved the original word count and did not introduce positional shifts in the text. The achieved substitution rates were 1.25%, 2.07%, and 3.20%, consistently below the target values. This undershoot is an inherent limitation of rule-based synonym substitution: not every eligible content word has a single-word synonym available in WordNet, so some selected replacements fail. The proportion of spam emails that received at least one synonym change was 69.6% at the light level, 81.6% at medium, and 89.1% at

heavy; the remaining emails consisted predominantly of short messages whose eligible words lacked available synonyms.

The third attack type was text dilution. A fixed list of professional and business vocabulary was compiled, including terms such as "meeting", "regards", "schedule", "report", "please", "attached", "update", "confirm", "agenda". For each spam email, the specified number of words was drawn from this list and inserted at random positions within the email body. The injection counts were set at 3, 7, and 12 words for the light, medium, and heavy levels respectively. Because injection is unconditional on word availability, all 3,424 spam emails were modified at every intensity level, and the achieved injection counts matched the targets exactly.

The complete adversarial evaluation framework therefore consisted of ten test conditions: one clean (unmodified) test set and nine adversarial variants. Both the TF-IDF + Logistic Regression baseline and the fine-tuned DistilBERT model were evaluated against all ten conditions using identical inputs, with the same classification metrics applied throughout. The achieved perturbation rates and modification counts for the nine adversarial conditions are summarised in Table 3.1.

Attack Type	**Intensity**	**Target Rate**	**Achieved Rate**	**Spam Modified**
Character Perturbation	Light	5%	8.27%	3,419 / 3,424 (99.85%)
Character Perturbation	Medium	10%	16.41%	3,422 / 3,424 (99.94%)
Character Perturbation	Heavy	20%	31.46%	3,422 / 3,424 (99.94%)
Synonym Substitution	Light	1%	1.25%	2,383 / 3,424 (69.60%)
Synonym Substitution	Medium	3%	2.07%	2,794 / 3,424 (81.60%)
Synonym Substitution	Heavy	5%	3.20%	3,049 / 3,424 (89.05%)
Text Dilution	Light	3 words	3 words	3,424 / 3,424 (100%)
Text Dilution	Medium	7 words	7 words	3,424 / 3,424 (100%)
Text Dilution	Heavy	12 words	12 words	3,424 / 3,424 (100%)

Table 3.1: Achieved perturbation rates and modification counts for the nine adversarial conditions

3.3 Testing

The system was verified across the four layers it comprises: the mail-security stack, the model pipelines, the adversarial generator, and the cascade-and-milter integration. Each layer was tested in isolation against the failure modes specific to it before the cascade was exercised end-to-end through the running system, with the integrated run reported in Section 3.3.4 providing the unifying test of behaviour at scale.

3.3.1 Mail-Security Verification

The mail stack was tested as a sealed system using black-box transaction tests at the SMTP and IMAPS boundaries. Postfix, Dovecot, OpenDKIM, and OpenDMARC were configured, not coded from scratch, so what needs verification is the protocol behaviour they produce under that configuration, not their internal logic. Three areas were exercised: the TLS handshakes that protect IMAP and submission, the authentication boundary on the submission service, and the outbound DKIM signing path that runs alongside the cascade milter.

Transport security was checked by opening IMAPS connections to TCP/993 with *openssl s_client* and forcing each TLS version in turn. TLS 1.2 negotiated *ECDHE-RSA-AES128-GCM-SHA256* and TLS 1.3 negotiated *TLS_AES_256_GCM_SHA384*, while TLS 1.0 and TLS 1.1 were refused at the protocol layer with *no protocols available*. The submission service on TCP/587 upgraded to TLS 1.3 with the same TLS 1.3 cipher when STARTTLS was issued, and the certificate chain shown in the openssl output confirmed that the server certificate (*CN=mailserver*) was issued by the local *Dissertation Root CA* mounted into the container. The *Verify return code: 21* reported alongside is the host trust store missing the local CA rather than a TLS defect, and is suppressed by passing *-CAfile* pointed at the local CA.

The submission boundary was checked from two directions. First, the host was confirmed not to expose plaintext IMAP: TCP/143 returned connection-refused for both IPv4 and IPv6, while TCP/993 was reachable as expected. Second, the submission service refused authentication before the channel was encrypted. The pre-STARTTLS *EHLO* response did not advertise *AUTH PLAIN* or *AUTH LOGIN* at all, and an attempted *AUTH LOGIN* before STARTTLS was rejected with *530 5.7.0 Must issue a STARTTLS command first*; *AUTH* only appeared in the EHLO response after the channel was encrypted. An unauthenticated message sent from an external address reached *RCPT TO* and was rejected with *554 5.7.1 ... Client host rejected: Access denied*, confirming that the docker-mailserver *master.cf* scopes SMTP AUTH to the submission path rather than to TCP/25.

Outbound DKIM signing and the milter integration on top of it were checked together. A test message sent through the submission service was delivered to the recipient mailbox and the delivered headers were captured directly from the maildir. The broader Postfix log captured during the diagnostic run recorded *TLSv1.3 with cipher TLS_AES_256_GCM_SHA384* on the SMTPSA hop for the message, providing the transport-side evidence. The delivered message itself carried a *DKIM-Signature* header with *d=mail.local* and *s=mail*, and the OpenDKIM milter logged a corresponding *DKIM-Signature field added (s=mail, d=mail.local)* entry against the same queue id, providing both header-side and server-side confirmation. The same delivered message also carried *X-Spam-Status*, *X-Spam-Score* and *X-Spam-Model: distilbert* headers, confirming that the cascade milter ran in the expected order and that its verdict reached the user mailbox. Sieve-driven routing into Junk was independently exercised by the cascade test described in Section 3.3.4.

Raw *openssl s_client* captures, EHLO transcripts and delivered-header captures from the recipient maildir are in *evidence/*.

3.3.2 ML Pipeline Verification

The two classifier pipelines were verified through a combination of run-time assertions and held-out evaluation. To detect overfitting in the baseline before final evaluation, five-fold stratified cross-validation was conducted on the training partition only, and the narrow gap between training and validation scores confirmed that the learned decision boundary did not depend on a particular subset of the data (Section 3.2.5). Run-time assertions were embedded in the DistilBERT notebook to verify that the row count and split sizes matched the baseline pipeline exactly, with class distribution preserved by stratified splitting on the same *random_state=42* seed; this eliminated the risk that environment differences between scikit-learn and the HuggingFace Trainer had silently introduced data drift between the two models (Section 3.2.6). Both pipelines reused the 80/20 stratified split with *random_state=42* so that final test results could be attributed to the model architecture rather than to data variation. The trained baseline (joblib) and DistilBERT (HuggingFace *save_pretrained*) artefacts were the same files consumed by the milter at runtime, so the verified models and the deployed models were guaranteed identical.

Beyond correctness, the cascade's inference latency was measured directly inside the spamfilter container after warmup, on the development host described in Section 3.2 (Apple M2, 16 GB RAM, Docker Desktop with the spamfilter container under Rosetta x86_64 emulation, single-threaded ML libraries per the threading-deadlock fix). Integration-level submission throughput was sustained at 5 messages per second across the 900-message

adversarial sample (Section 3.3.4) without errors, queueing failures or timeouts. The figures in Table 3.2 are preliminary single-submission measurements; concurrent multi-tenant load, CPU and RAM under sustained traffic, and burst-arrival behaviour are not characterised here.

Path	Cold-start load	Warmup call	Steady-state per message
Baseline (TF-IDF + Logistic Regression)	0.35 s	0.9 ms	0.2–0.3 ms
DistilBERT (escalated path)	2.19 s	240 ms	30–50 ms

Table 3.2: Cascade per-path latency (steady state after warmup) and one-time startup costs.

3.3.3 Adversarial Generator Verification

The adversarial generator was verified by comparing target and achieved perturbation rates across all nine conditions (Table 3.1).

3.3.4 Milter Integration Verification

Across the nine adversarial conditions and 900 messages submitted through the live mail stack, 446 of 450 spam were routed to Junk and 450 of 450 ham to Inbox, a system-level accuracy of 99.6% that tracks the offline DistilBERT figures within sampling noise (Table 3.3). The remainder of this subsection sets out how that figure was produced, beginning with the white-box branch coverage suite and extending to the stratified per-condition run.

End-to-end verification of the cascade was performed with a deliberately small, white-box test suite designed to exercise every branch of *classify_email()* rather than to repeat the broader evaluation reported in Section 4.2. Three messages were submitted through the authenticated submission service via *scripts/send_milter_tests.sh*: an obvious ham (informal personal mail) and an obvious spam (a high-pressure prize-claim message), which exercised the baseline-only path; and a borderline phishing attempt designed to land in the cascade's uncertainty band, which exercised the DistilBERT escalation path. Branch coverage was confirmed by inspecting the structured log line emitted by the milter for each message; the cascade-escalation branch produced lines of the form *status=Yes score=0.9990 model=distilbert*. Routing was confirmed by inspecting the recipient's Roundcube mailbox: ham remained in Inbox, while the spam and borderline messages were routed into Junk by the Pigeonhole sieve rule.

The white-box suite was extended to a stratified random sample of 100 messages per adversarial condition (50 spam, 50 ham, fixed seed for reproducibility), submitted through

authenticated SMTP and read back over IMAPS via *scripts/send_adversarial_e2e.py*. Each message was tagged with a per-run identifier so concurrent invocations did not contaminate each other; per-message verdicts (*X-Spam-Status*, score, model) and the resulting folder placement are recorded in *evaluation/end_to_end/*, with one CSV per condition. The script's loop submitted each message via authenticated SMTP, waited for the tagged message to appear in the recipient's mailbox over IMAPS, parsed its *X-Spam-** headers and recorded the folder it landed in before submitting the next, so the system-level signal compared against the spam/ham label was the user-visible mailbox routing rather than the model verdict alone.

Condition	Spam → Junk	Ham → Inbox	System acc	Dataset DistilBERT acc
Character (light)	50/50	50/50	100%	99.75%
Character (medium)	50/50	50/50	100%	99.79%
Character (heavy)	50/50	50/50	100%	99.79%
Synonym (light)	50/50	50/50	100%	99.57%
Synonym (medium)	50/50	50/50	100%	99.61%
Synonym (heavy)	50/50	50/50	100%	99.58%
Dilution (light)	50/50	50/50	100%	99.06%
Dilution (medium)	48/50	50/50	98%	97.95%
Dilution (heavy)	48/50	50/50	98%	96.81%
Total	**446/450**	**450/450**	**99.6%**	—

Table 3.3: End-to-end results: 100 messages per adversarial condition through the live mail stack, with dataset-level DistilBERT accuracy for comparison.

The four false negatives concentrate in dilution medium and heavy, matching the dataset-level finding that text dilution is the differentiating attack against the cascade. The integrated run therefore reproduces the offline evaluation rather than diverging from it: the cascade and the Sieve routing on top of it preserve the model-level result through to mailbox placement.

The same suite, run before the threading-deadlock fix described in Section 3.2.8, demonstrated the diagnostic value of integration-level testing. Four wedged copies of the borderline message were delivered to Inbox without spam headers under *milter_default_action = accept*, exposing both the deadlock and the routing failure that fail-open behaviour created in the silent-deadlock window. The diagnosis and fix are described in Section 3.2.8.

3.3.5 Coverage Summary and Residual Risk

Across the four product layers, verification eliminated specific kinds of error: TLS and cipher misconfiguration, plaintext-auth drift, AUTH-scope leakage, and missing outbound DKIM

signing in the mail-security stack; train/test contamination, mismatched preprocessing, and overfitting in the ML pipelines; miscalibrated attack intensity, eligibility-filter regressions and word-count drift in the adversarial generator; and cascade-branch logic faults, malformed header injection, broken Sieve routing and the integration-level threading deadlock in the milter. The principal residual risks are functionality characteristics that were not measured: inference latency, throughput and resource consumption under sustained traffic (Section 4.1, taken up in Section 5.1). Other untested dimensions include inbound DMARC/SPF (no external sender), alternative MUA behaviour, threshold sensitivity to a different ham/spam mix and external interoperability beyond this single host.

4. Discussion and Evaluation

4.1 Evaluation of the Product

The product is the hardened local email system and the cascade ML spam filter taken as a whole, evaluated against the aims and objectives set out in Section 1.2 and the requirements set out in Section 2.3.

On the functional side, the mail stack delivered authenticated submission, encrypted mailbox access, SPF, DKIM and DMARC enforcement, and fail2ban-based brute-force protection (ToR objective 1), verified end-to-end through the captures referenced in Section 3.3.1. Integration of the cascade and Sieve routing was further verified at scale in Section 3.3.4: across 900 messages spanning the nine adversarial conditions, system-level accuracy tracked the dataset-level DistilBERT figures within sampling noise, with no false-positive routings of ham to Junk. The dataset and preprocessing pipeline, both classifiers, and the standard-metric evaluation that compared them on identical splits (ToR objectives 2–5 and 7) are reported in Sections 3.2 and 4.2. The adversarial test suite covering character, synonym, and dilution perturbations at three intensities each (ToR objective 6) is described in Section 3.2.9, and integration as a Postfix milter with X-Spam-* header injection and Sieve quarantine into Junk (ToR objective 8) is described in Section 3.2.7. The cascade architecture extends the original single-transformer ToR into a confidence-routed two-stage classifier (Section 3.2.7). Reproducibility (non-functional requirement) was met through versioned container tags, fixed random seeds, and identical model artefacts at training and deployment time, eliminating a class of train/deploy drift defect that would otherwise be invisible. The integration-level threading deadlock encountered during testing (Section 3.2.8) is itself a qualified strength: the three-layer fix was verified, with residual fail-open behaviour bounded by an explicit per-call timeout rather than the default 180-second *milter_ content_ timeout*.

The product was partial on one ToR objective and exposed two further weaknesses worth critical comment. ToR objective 9, systematic functionality benchmarking, was partially delivered: per-message inference latency and integration throughput are reported in Section

3.3.2, but CPU and RAM under sustained traffic, multi-tenant load, and burst-arrival behaviour are not characterised. The cascade threshold of 0.90 was a reasoned starting point informed by clean-data accuracy rather than an empirically tuned value, leaving threshold sensitivity to a different ham/spam mix unknown. The fail-open policy preserves mail flow but routes unclassified messages to the user inbox during any silent failure window; the three-layer fix closes the specific deadlock but does not change the policy. The product was exercised on a single host with a single dataset (Enron-Spam, predating contemporary LLM-generated spam), so external interoperability and behaviour under modern threat distribution are extrapolated rather than tested.

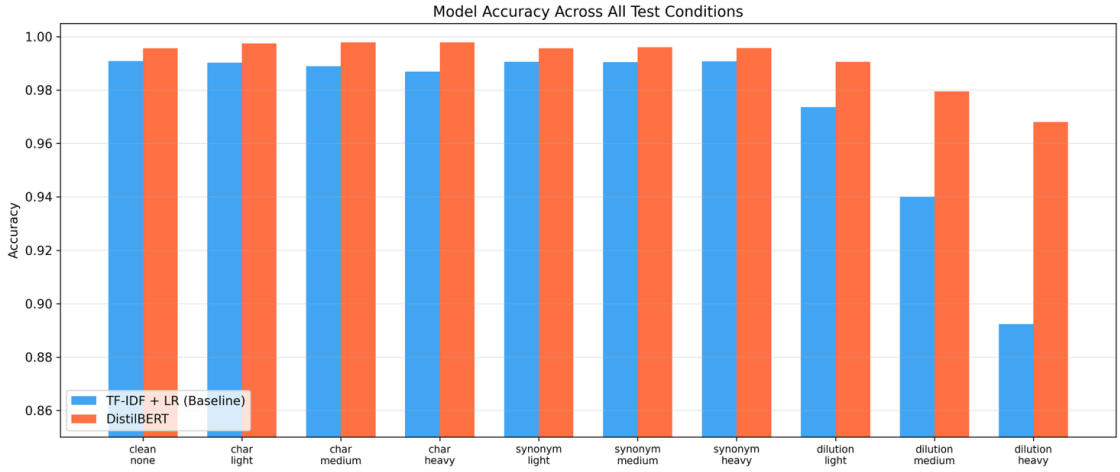
Alternative approaches were rejected for documented reasons: Postfix's *content_filter* operates post-queue and breaks header tagging (Section 3.2.7); an external SMTP proxy duplicates transport handling Postfix already provides; lazy-loading DistilBERT shifts cold-start latency to the first escalation (Section 3.2.7); client-side Sieve filtering applies per-MUA (Section 3.2.7); and model-specific adversarial attacks conflate architecture with attack design. Adversarial training and ensembling remain promising directions taken up in Section 5.1.

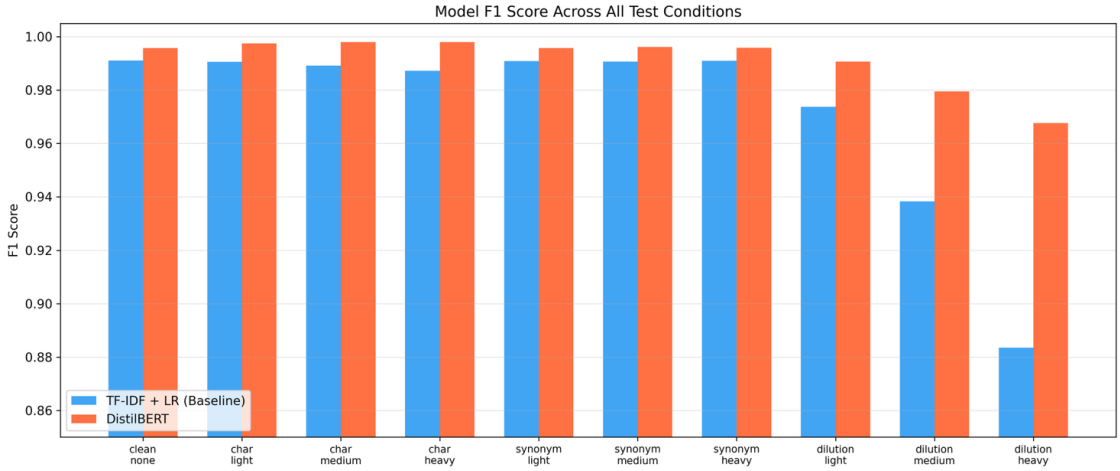
Narrower limitations of the supporting investigation are addressed in Section 4.2.3; forward-looking recommendations that would strengthen the product against the weaknesses surfaced above are taken up in Section 5.1.

4.2 Adversarial Evaluation

As part of evaluating the cascade architecture, both classifier components were measured on clean and adversarial data. On the unmodified test set of 6,733 emails, the TF-IDF + Logistic Regression baseline recorded 99.09% accuracy, 98.53% precision, 99.71% recall, an F1 score of 99.11%, and a ROC-AUC of 99.91%; the fine-tuned DistilBERT model recorded 99.55% accuracy, 99.68% precision, 99.45% recall, an F1 of 99.56%, and a ROC-AUC of 99.98%. The 0.46 percentage-point clean-data advantage is consistent with published results for transformer-based classifiers over traditional approaches (Garrido-Merchán et al., 2023; Alhuzali et al., 2025), and exceeds the BERT (98.99%) and RoBERTa (99.08%) figures Alhuzali et al. (2025) reported on comparable datasets. Both models also surpass the Metsis et al. (2006) Naive Bayes recall of 97.53% and the Kshirsagar et al. (2025) SVM-with-TF-IDF accuracy of 97.7% on the same Enron-Spam corpus, which the present baseline attributes to bigram features, sublinear TF scaling, and the discriminative power of Logistic Regression relative to generative classifiers. The DistilBERT result likely reflects the favourable vocabulary separation in the Enron corpus identified during exploratory analysis

(Section 3.2.4). The strong clean-data performance of both models establishes a credible baseline from which to measure adversarial degradation: any decline under attack can be interpreted as a genuine effect of the perturbation rather than an artefact of weak initial classification.





4.2.1 Character and Synonym Attacks

Character-level perturbation produced minimal degradation in both models, but the pattern of responses differed. The baseline experienced a steady, monotonic decline as perturbation intensity increased: accuracy fell from 99.09% on clean data to 99.03% at light (8.27% character modification), 98.90% at medium (16.41%), and 98.69% at heavy (31.46%), representing a cumulative drop of 0.40 percentage points. The error increase was driven entirely by additional false negatives (spam misclassified as ham), rising from 10 on clean data to 37 at heavy, while false positives remained constant at 51 across all conditions. This asymmetry indicates that character edits disrupted spam-indicative token matches without introducing

new spam-like features into ham emails, consistent with the token-frequency dependence of TF-IDF representations identified in Section 3.2.5.

DistilBERT showed no degradation under character perturbation. Accuracy was 99.75% at light, 99.79% at medium, and 99.79% at heavy, all marginally above the clean-data figure of 99.55%. False negatives decreased from 19 on clean data to 2 at medium and heavy intensity. DistilBERT splits misspelled or altered words into smaller subword pieces using its WordPiece tokenizer, rather than treating them as unknown tokens. A perturbed word therefore still reaches the model as familiar fragments, giving the transformer a natural resistance to character-level edits without any adversarial training.

Synonym substitution had negligible impact on both models across all three intensity levels. The baseline maintained accuracy between 99.05% and 99.08% (within 0.04 percentage points of clean performance), while DistilBERT maintained accuracy between 99.57% and 99.61%. Neither model exhibited a meaningful degradation trend as substitution intensity increased from light (1.25% of words replaced) to heavy (3.20%).

Two factors explain this result. First, the achieved substitution rates were substantially below the target rates (3.20% achieved versus 5% target at heavy), because WordNet frequently lacks single-word synonyms for eligible content words, particularly the domain-specific vocabulary common in spam emails. At these low replacement rates, fewer than 4% of words in the average spam email were altered, providing insufficient perturbation to shift either model's classification decision. Second, the synonyms that were successfully applied were, by definition, semantically related to the original words; a model that has learned to associate certain semantic fields with spam (financial terms, urgency language, commercial offers) may recognise the replacement as belonging to the same semantic neighbourhood. Rule-based synonym substitution at these rates does not constitute a credible evasion threat against either model; context-aware substitution methods using masked language models would likely produce stronger results (Morris et al., 2020b).

4.2.2 Text Dilution Attack

Text dilution was the only attack that produced substantial and clearly differentiated degradation between the two models, as illustrated in Figure 4.1. The baseline's accuracy declined from 99.09% on clean data to 97.36% at light intensity (3 benign words injected), 94.00% at medium (7 words), and 89.23% at heavy (12 words), representing a total drop of 9.86 percentage points. This was accompanied by a sharp rise in false negatives, from 10 on clean data to 674 at heavy, meaning that 19.7% of spam emails were misclassified as legitimate after the injection of just 12 professional-vocabulary words. The decline was monotonic and

accelerating: each additional tier of injection produced a larger marginal accuracy loss than the previous one, indicating that the baseline's decision boundary was being progressively overwhelmed as benign tokens diluted the relative concentration of spam-indicative features.

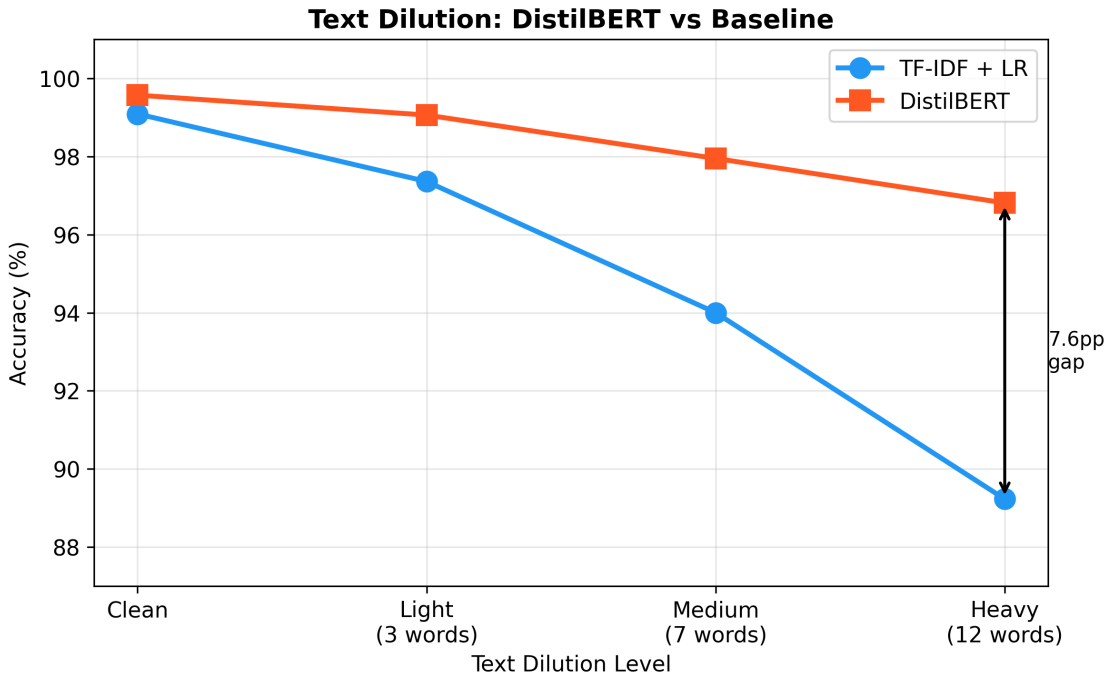


Figure 4.1: Accuracy of TF-IDF + Logistic Regression baseline and DistilBERT under text dilution at light, medium and heavy intensities.

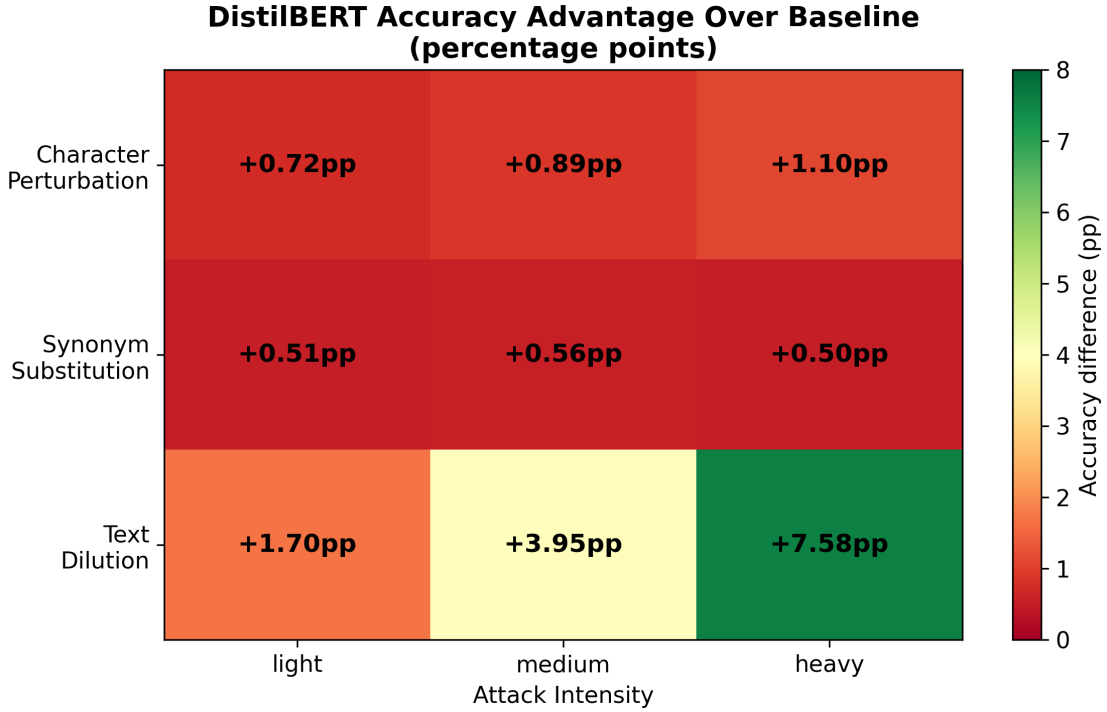
DistilBERT exhibited significantly greater resilience. Accuracy fell from 99.55% to 99.06% at light, 97.95% at medium, and 96.81% at heavy, a total drop of 2.74 percentage points. At heavy intensity, DistilBERT retained 96.81% accuracy compared with the baseline's 89.23%, an absolute advantage of 7.58 percentage points. Expressed as relative degradation, the baseline lost 3.6 times more accuracy than DistilBERT under identical attack conditions.

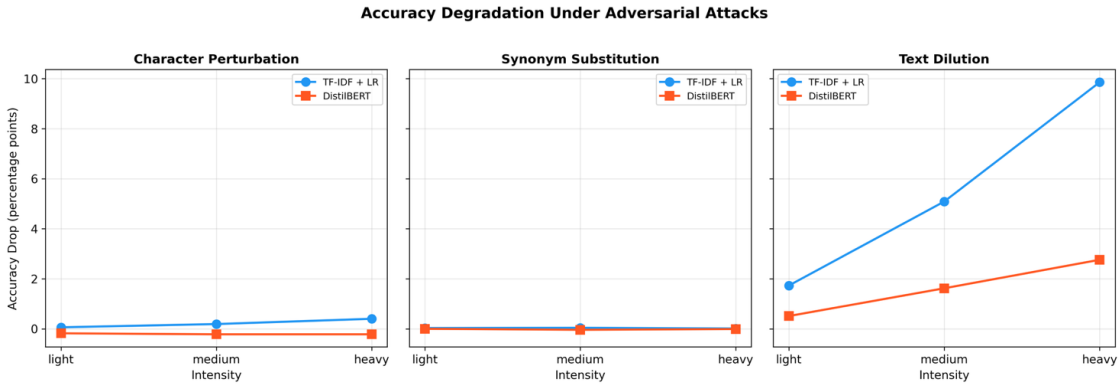
The mechanism follows directly from the architectural difference between the two models. TF-IDF classification is driven by the statistical distribution of token weights across the feature vocabulary; when benign words such as "meeting", "regards" and "schedule" are injected, they introduce high-weighted ham-associated features that shift the Logistic Regression decision function toward the legitimate class, and the effect compounds with the number of injected words because TF-IDF treats all tokens as independent and equally positioned. DistilBERT's self-attention mechanism, by contrast, computes representations in which each token's meaning is conditioned on its surrounding context. Benign words inserted at random positions lack coherent contextual relationships with the existing spam content, and the

attention mechanism assigns lower influence to tokens that do not integrate meaningfully, so the injected words have a reduced capacity to shift classification.

This finding is consistent with Kuchipudi et al. (2020), who demonstrated that ham word injection is an effective evasion technique against traditional spam classifiers, and extends their analysis by providing a direct comparison showing that contextual models are substantially more resistant to the same manipulation. The result also aligns with the broader framing of Hotoğlu et al. (2025), who identified content injection as a growing dimension of spam filter evasion and argued that robustness evaluation should encompass injection-based attacks alongside token-level edits.

Across the three attack families, the supporting investigation supports a qualified affirmative answer to its question: DistilBERT retains accuracy better than the TF-IDF + Logistic Regression baseline under realistic text manipulation, with the advantage concentrated in content-injection. Under character perturbation DistilBERT was effectively immune while the baseline experienced modest measurable degradation; under synonym substitution neither model was materially affected; under heavy dilution DistilBERT retained nearly four times more accuracy than the baseline. This pattern carries through to the integrated system: in the end-to-end run reported in Section 3.3.4, all four false negatives observed across the 900-message stratified sample fell in dilution medium and dilution heavy, with every other condition routed correctly.





4.2.3 Limitations

Several limitations constrain the generalisability of these findings. The adversarial attacks were model-agnostic; model-specific attacks such as TextFooler (Jin et al., 2020) and BERT-Attack (Li et al., 2020) would likely produce larger degradation in both models and may alter the relative comparison. The study evaluated a single dataset (Enron-Spam), and the pronounced vocabulary separation between ham and spam in this corpus may inflate clean-data accuracy and reduce the apparent effect of perturbations that do not bridge this separation; replication on more heterogeneous corpora would strengthen the external validity of the findings. The synonym substitution attack was further constrained by the limitations of rule-based WordNet lookup, which restricted achieved substitution rates to below 4% of words; context-aware substitution methods using masked language models would likely produce higher replacement rates and more semantically coherent perturbations. A related limitation is that the suite contains no semantically coherent rewrites: text dilution injects benign tokens but does not preserve sentence-level fluency, so adversarial paraphrase produced by an LLM, in which surface form is rewritten while spam intent is retained, is not captured (Hotoğlu et al., 2025). All metrics derive from a single training run per model with a fixed seed, so initialisation variance is not characterised; the 0.46 percentage-point clean-data difference in particular sits within the range that re-seeded runs could plausibly close, while the larger gap under heavy text dilution is less susceptible. The adversarial suite measures classifier-level evasion but does not verify that perturbed messages remain functional for an attacker: at heavy character-perturbation, the message may be too degraded for a human recipient to act on, so apparent evasion may not correspond to a credible attack. The cascade also operates on extracted body text alone, whereas production filters combine body-text signals with envelope, authentication and reputation features that the surrounding mail stack produces but does not pass to the classifier; the present results therefore reflect only a subset of the signals a production filter would draw upon. Finally, no adversarial training

or defensive preprocessing was applied to either model; the results therefore reflect default behaviour under attack rather than the performance ceiling achievable through targeted hardening. These limitations are discussed further in the recommendations for future work (Section 5.1).

4.3 Evaluation of the Project Processes

Adversarial robustness research in spam detection has largely treated the classifier as an isolated artefact, evaluated against perturbed datasets in benchmark settings. This project took a different position. Embedding the AI spam filter inside a hardened end-to-end mail stack (authenticated submission, transport security, full SPF/DKIM/DMARC enforcement, milter integration and Sieve quarantine) let the work measure classifier behaviour as one component of a layered system rather than as a standalone benchmark. That product-led framing shaped the methodology and the standards for what counted as a meaningful result.

Eight of the ten Terms of Reference objectives were met as planned; one shifted in execution and one was partially delivered. The shift concerned the adversarial suite. Unicode homoglyphs and leet substitution were originally listed, but during execution text dilution replaced them as a more discriminating attack, and the simpler implementation gave more confidence in correct verification within the available time (Section 4.2.2). Objective 9, systematic functionality benchmarking under load, was partially delivered: single-submission latency and integration throughput are reported in Section 3.3.2, but a controlled multi-tenant pass under sustained traffic was not performed and is taken up in Section 5.1.

The single largest unforeseen issue was the threading deadlock described in Section 3.2.8, which consumed several days of diagnostic work the ToR had not anticipated. With hindsight, a one-week buffer between practical completion and final write-up would have absorbed unplanned diagnostic work of this kind. Whether the three-layer fix is the correct architectural solution or a defensible workaround that behaves correctly in this particular instance remains an open question, integrating ML inference into a libmilter worker is not well-documented, and that uncertainty is itself worth noting.

Practical implementation and the corresponding written sections were developed in parallel rather than sequentially, which preserved logs, and overall continuity that would otherwise have been reconstructed retrospectively.

The project also raised standard dual-use considerations. The perturbation techniques used to evaluate filter accuracy under attack could in principle be used to evade filters, however, the attacks are model-agnostic, build on already-published methods, were never

executed against a third-party system, and the system itself is local and not internet-exposed. The dataset is publicly released and contains no personal data. Configurations, random seeds and source code are versioned.

The most substantive lesson concerned the gap between notebook performance and runtime behaviour. A model that completed inference in under fifty milliseconds in isolation could still silently fail inside a libmilter worker because of an interaction between Rosetta emulation, libgomp and lazy thread pool initialisation.

A further instance surfaced when I extended the end-to-end test to the adversarial corpus. On the first run, ham messages misclassified at 100% because the Python email library encoded the body in a way the milter's body extractor could not decode. Switching the test client to plain-text body construction restored agreement with the offline evaluation. As someone new to ML, I had not anticipated that the format of a classifier's input could differ between a notebook and a running mail server in ways that change every verdict, and working through the diagnostic was a substantive lesson for me on integration testing.

A second lesson concerned evaluation realism. Model-agnostic attacks applied at random positions are conservative, and the literature indicates that model-specific attacks would produce sharper degradation in both classifiers (Jin et al., 2020). The most transferable skill from this project, alongside the ML implementation work itself, is reading across a focused literature and forming an independent judgement, since this is my first academic writing produced at this scale.

With hindsight, three notable changes would strengthen the design. First, augmenting or replacing Enron-Spam with a more recent corpus, ideally one containing AI-generated content, since contemporary spam increasingly originates from generative models (Hao et al., 2025) and Enron predates that distribution shift. Second, including at least one model-specific attack alongside the model-agnostic suite, to give a stricter accuracy-drop ceiling and a more direct comparison with the existing literature. Third, scoping a small evaluation of a tuned large language model as a third classifier; that direction remains underexplored in the spam-filter literature and would align the project more closely with the threat profile it set out to address. Enron-Spam was selected because it supports comparability with prior work, which was defensible at the time, but the field has moved.

5. Conclusion and Recommendations

This project built a hardened local email system paired with a cascade AI spam filter, evaluated as a working product. The four product layers, the mail-security stack, the model pipelines, the adversarial generator, and the milter integration, were verified end-to-end through the testing approach described in Section 3.3.

The principal aim stated in Section 1.2 was met. The product objectives set out in the same section were delivered with one shift in scope and one partial outcome: the mail stack provided authenticated submission, encrypted mailbox access, and full SPF, DKIM and DMARC enforcement (Sections 3.2.1 to 3.2.3); the two classifiers were trained under a single dataset split, preprocessing pipeline, and evaluation protocol (Sections 3.2.4 to 3.2.6); the cascade was integrated as a Postfix milter operating in the SMTP transaction (Section 3.2.7) and escalated to DistilBERT only when baseline confidence fell below threshold (Section 3.2.7); the verdict travelled through standard *X-Spam-** headers and was routed to Junk by a server-side Dovecot Sieve rule (Section 3.2.7); and the adversarial test suite covered character-level, word-level, and content-level perturbations at three intensities each (Section 3.2.9). The shift concerned the adversarial suite, where text dilution replaced the originally-planned Unicode homoglyph and leet substitution attacks during execution (Section 4.3). Objective 9, systematic functionality benchmarking under load, was partially delivered and is taken up in Section 5.1. The non-functional requirements of reproducibility, verifiability, modularity, and security (Section 2.3) were each met as described in the same chapters and confirmed through the four-layer testing approach (Section 3.3).

The supporting research aim concerning the comparative behaviour of the two classifier families under adversarial attack was also addressed. Both models performed strongly on clean data, with 99.09% accuracy for the baseline and 99.55% for DistilBERT. Character perturbation produced minimal degradation in either model, and rule-based synonym substitution had negligible effect at the achieved replacement rates. Text dilution was the differentiating attack: at heavy intensity the baseline fell to 89.23% accuracy while DistilBERT retained 96.81%, a 7.58 percentage-point gap that follows directly from self-attention discounting contextually

disconnected tokens, whereas frequency-based features treat every token as independent and equally weighted (Section 4.2). Three product weaknesses identified in Section 4.1, threshold sensitivity, the fail-open policy, and single-host extrapolation, temper the strength of these conclusions. The principal residual gap, acknowledged in Section 4.1, is that systematic CPU and RAM benchmarking under sustained traffic was not performed; preliminary latency and throughput figures are in Section 3.3.2, with full functionality characterisation taken up in Section 5.1.

Across nine adversarial conditions and 900 messages through the live mail stack, the integrated system routed 446 of 450 spam to Junk and 450 of 450 ham to Inbox; system-level accuracy was 99.6%, tracking offline DistilBERT within sampling noise. Within the bounds of the project, that is the standard the work was held to and met: the AI spam filter measured as a verified component of a hardened mail stack rather than as a detached classifier. The unfinished piece is sustained-load functionality, picked up in the recommendations that follow.

5.1 Recommendations

Operationally, the spam filter is a system component first and a model choice second. Model selection should follow from the anticipated threat profile rather than from clean-data accuracy: where the dominant evasion is content injection or statistical-profile manipulation, DistilBERT's accuracy-retention premium justifies its higher computational cost, while a well-tuned TF-IDF baseline remains adequate where the threat is dominated by typographical obfuscation or minor lexical substitution. The cascade architecture deployed here, baseline first and transformer escalated only when baseline confidence falls below threshold, is a deployment recommendation in itself: it confines the cost of the larger model to the cases that need it, while preserving the accuracy retention measured in Section 4.2.

Equally, the right unit of verification is the running system, not the classifier in isolation. Two integration faults that mattered in this project, the threading deadlock under concurrent submissions (Section 3.2.8) and a body-encoding mismatch between the offline test client and the milter (Section 4.3), were invisible to classifier-only testing and only surfaced when the cascade was exercised through the live mail stack. End-to-end submission and IMAPS read-back testing, with verdict headers and Sieve-driven folder placement both checked, is the standard at which any milter-integrated classifier should be assessed; the 900-message run reported in Section 3.3.4 is what that standard looks like in practice.

Two methodological recommendations emerge for similar evaluations. Reporting degradation curves across attack intensities, rather than single-point accuracies, captures divergence

that lighter conditions alone would miss; in this work, the gap between the two models was only visible at the heavy condition, and a single-point comparison would have produced a misleading conclusion. Pairing a model-agnostic suite with a model-specific attack such as TextFooler (Jin et al., 2020) or BERT-Attack (Li et al., 2020) is recommended for tighter accuracy-drop ceilings.

Six directions would extend this project beyond its undergraduate scope, following from the limitations identified in Section 4.2.3 and the partial functionality finding in Section 3.3.2. First, replication on a more contemporary corpus is needed because Enron-Spam predates the recent shift in spam composition, with at least 51% of spam in a 2025 corpus reported as LLM-generated (Hao et al., 2025). Second, layered defences such as adversarial training, ensembling, and paired transport-level controls should be evaluated against the same benchmark, since the architectural accuracy retention reported here is a floor rather than a ceiling. Third, large language models warrant evaluation along two axes the present design omits: as an adversarial generation mechanism producing contextually coherent rewrites that text dilution only crudely approximates, and as a third classifier family currently underexplored in the spam-filter literature. A controlled comparison using a fixed instruction-tuned model, a fixed prompt, and seeded sampling would test whether DistilBERT's resistance to random word injection extends to fluent end-to-end paraphrase. Fourth, systematic functionality benchmarking covering inference latency, throughput, and resource consumption under load should be conducted within the milter pipeline. Fifth, cascade behaviour should be evaluated under degraded auth-chain or transport-security conditions, since real mail systems experience configuration drift and partial outages. Sixth, more compact transformer variants such as TinyBERT (Jiao et al., 2020) warrant evaluation for resource-constrained deployment where DistilBERT's footprint may be problematic.

References

Adnan, M., Imam, M.O., Javed, M.F. and Murtza, I. (2024) ' Improving spam email classification accuracy using ensemble techniques: a stacking approach', *International Journal of Information Security*, 23, pp. 505–517. doi: 10.1007/s10207-023-00756-1.

Al-augby, S., Alyasiri, H., Ghalib Abdulkadhim, F. and Ch. Oleiwi, Z. (2025) ' A stacked ensemble classifier for email spam detection via an evolutionary algorithm', *Mesopotamian Journal of CyberSecurity*, 5(2), pp. 657–670. doi: 10.58496/MJCS/2025/039.

Alhuzali, A., Alloqmani, A., Aljabri, M. and Alharbi, F. (2025) ' In-depth analysis of phishing email detection: evaluating the performance of machine learning and deep learning models across multiple datasets', *Applied Sciences*, 15(6), 3396. doi: 10.3390/app15063396.

Ashiq, M.I., Li, W., Fiebig, T. and Chung, T. (2023) 'You've Got Report: Measurement and Security Implications of DMARC Reporting', *Proceedings of the 32nd USENIX Security Symposium*, Anaheim, CA, USA, 9–11 August.

Ashiq, M.I., Fiebig, T. and Chung, T. (2025) 'Unraveling the Complexities of MTA-STS Deployment and Management in Securing Email', *Proceedings of the 2025 ACM Internet Measurement Conference (IMC '25)*, Madison, WI, USA. ACM. doi:10.1145/3730567.3732916.

Bhowmick, A. and Hazarika, S.M. (2018) ' E-Mail Spam Filtering: A Review of Techniques and Trends', in *Advances in Electronics, Communication and Computing*, Lecture Notes in Electrical Engineering. Singapore: Springer. doi: 10.1007/978-981-10-4765-7_61.

Crocker, D., Hansen, T. and Kucherawy, M. (2011) *DomainKeys Identified Mail (DKIM) Signatures*. RFC 6376. RFC Editor. doi:10.17487/RFC6376.

Czybik, S., Horlboge, M. and Rieck, K. (2023) 'Lazy Gatekeepers: A Large-Scale Study on SPF Configuration in the Wild', in *Proceedings of the 2023 ACM Internet Measurement Conference (IMC '23)*, Montreal, QC, Canada, 24–26 October. New York, NY: ACM, pp. 344–355. doi:10.1145/3618257.3624827.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019) ' BERT: pre-training of deep bidirectional transformers for language understanding', *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Minneapolis, MN, pp. 4171–4186. doi: 10.18653/v1/N19-1423.

docker-mailserver project (n.d.) *target/postfix/main.cf: Postfix configuration template shipped with docker-mailserver*. docker-mailserver source repository. Available at: https://github.com/docker-mailserver/docker-mailserver/blob/master/target/postfix/main.cf (Accessed: 30 April 2026).

Durumeric, Z., Adrian, D., Mirian, A., Kasten, J., Bursztein, E., Lidzborski, N., Thomas, K., Eranti, V., Bailey, M. and Halderman, J.A. (2015) 'Neither Snow Nor Rain Nor MITM... An Empirical Analysis of Email Delivery Security', *Proceedings of the 2015 ACM Internet Measurement Conference (IMC '15)*, Tokyo, Japan. ACM. doi:10.1145/2815675.2815695.

Free Software Foundation (n.d.) *GCC OpenMP runtime library (libgomp): environment variables*. Available at: https://gcc.gnu.org/onlinedocs/libgomp/Environment-Variables.html (Accessed: 27 April 2026).

Gao, J., Lanchantin, J., Soffa, M.L. and Qi, Y. (2018) ' Black-box generation of adversarial text sequences to evade deep learning classifiers', *2018 IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, pp. 50–56. doi:10.1109/SPW.2018.00016.

Garg, S. and Ramakrishnan, G. (2020) 'BAE: BERT-based Adversarial Examples for Text Classification', in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pp. 6174–6181 (Online). doi:10.18653/v1/2020.emnlp-main.498.

Garrido-Merchán, E.C., Gozalo-Brizuela, R. and González-Carvajal, S. (2023) ' Comparing BERT against traditional machine learning models in text classification', *Journal of Computational and Cognitive Engineering*, 2(4), pp. 352–356. doi: 10.47852/bonviewJCCE3202838.

Gu, Z., Hu, W. and Liu, J. (2021) ' Marginal attacks of generating adversarial examples for spam filtering', *Chinese Journal of Electronics*, 30(4), pp. 595–602. doi: 10.1049/cje.2021.05.001.

Hao, W., Tran, V., Rideout, V., Wang, Z., Dasbach-Prisk, A., Afifi, M.H., Yang, J., Katz-Bassett, E., Ho, G. and Cidon, A. (2025) ' Do spammers dream of electric sheep? Characterizing the prevalence of LLM-generated malicious emails', *Proceedings of the 2025 ACM Internet Measurement Conference (IMC '25)*, Madison, WI, 28–31 October. New York: ACM. doi: 10.1145/3730567.3732922.

Hoffman, P. (2002) *SMTP Service Extension for Secure SMTP over Transport Layer Security*.

Request for Comments 3207. RFC Editor. doi:10.17487/RFC3207.

Hotoğlu, E., Sen, S. and Can, B. (2025) 'A Comprehensive Analysis of Adversarial Attacks against Spam Filters', *arXiv* preprint, arXiv:2505.03831. doi:10.48550/arXiv.2505.03831.

Jamal, S., Wimmer, H. and Sarker, I.H. (2024) ' An improved transformer-based model for detecting phishing, spam and ham emails: a large language model approach', *Security and Privacy*, 7(3), e402. doi: 10.1002/spy2.402.

Jáñez-Martino, F., Alaiz-Rodríguez, R., González-Castro, V., Fidalgo, E. and Alegre, E. (2023) ' A review of spam email detection: analysis of spammer strategies and the dataset shift problem', *Artificial Intelligence Review*, 56(2), pp. 1145–1173. doi: 10.1007/s10462-022-10195-4.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F. and Liu, Q. (2020) ' Tiny-BERT: distilling BERT for natural language understanding', in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, pp. 4163–4174. doi: 10.18653/v1/2020.findings-emnlp.372.

Jin, D., Jin, Z., Zhou, J.T. and Szolovits, P. (2020) ' Is BERT really robust? A strong baseline for natural language attack on text classification and entailment', *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05), pp. 8018–8025. doi: 10.1609/aaai.v34i05.6311.

Kitterman, S. (2014) *Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1*. RFC 7208. RFC Editor. doi:10.17487/RFC7208.

Klensin, J. (2008) *Simple Mail Transfer Protocol*. RFC 5321. RFC Editor. doi:10.17487/RFC5321.

Kshirsagar, M., Rathi, V. and Ryan, C. (2025) ' Meta-learner-based frameworks for interpretable email spam detection', *Frontiers in Artificial Intelligence*, 8, 1569804. doi: 10.3389/frai.2025.1569804.

Kucherawy, M. and Zwicky, E. (2015) *Domain-based Message Authentication, Reporting, and Conformance (DMARC)*. RFC 7489. RFC Editor. doi:10.17487/RFC7489.

Kuchipudi, B., Nannapaneni, R.T. and Liao, Q. (2020) ' Adversarial machine learning for spam filters', *Proceedings of the 15th International Conference on Availability, Reliability and Security (ARES '20)*. New York: ACM. doi: 10.1145/3407023.3407079.

Li, L., Ma, R., Guo, Q., Xue, X. and Qiu, X. (2020) 'BERT-ATTACK: Adversarial Attack Against BERT Using BERT', in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pp. 6193–6202 (Online). doi:10.18653/v1/2020.emnlp-main.500.

Manita, G., Chhabra, A. and Korbaa, O. (2023) ' Efficient e-mail spam filtering approach combining Logistic Regression model and Orthogonal Atomic Orbital Search algorithm', *Applied Soft Computing*, 144(4), 110478. doi: 10.1016/j.asoc.2023.110478.

Margolis, D., Risher, M., Ramakrishnan, B., Brotman, A. and Jones, J. (2018) *SMTP MTA Strict Transport Security (MTA-STS)*. Request for Comments 8461. RFC Editor. doi:10.17487/RFC8461.

Metsis, V., Androutsopoulos, I. and Paliouras, G. (2006) ' Spam filtering with Naive Bayes – which Naive Bayes?', *3rd International Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA.

Moore, K. and Newman, C. (2018) *Cleartext Considered Obsolete: Use of Transport Layer Security (TLS) for Email Submission and Access*. RFC 8314. RFC Editor. doi:10.17487/RFC8314.

Moriarty, K. and Farrell, S. (2021) *Deprecating TLS 1.0 and TLS 1.1*. RFC 8996. RFC Editor. doi:10.17487/RFC8996.

Morris, J., Lifland, E., Yoo, J.Y., Grigsby, J., Jin, D. and Qi, Y. (2020a) 'TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP', in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, pp. 119–126 (Online). doi:10.18653/v1/2020.emnlp-demos.16.

Morris, J.X., Lifland, E., Lanchantin, J., Ji, Y. and Qi, Y. (2020b) ' Reevaluating adversarial examples in natural language', in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Stroudsburg, PA: Association for Computational Linguistics, pp. 3829–3839. doi: 10.18653/v1/2020.findings-emnlp.341.

Postfix Project (n.d.b) *master(5): Postfix master process configuration file format*. Postfix Documentation.

Postfix Project (n.d.e) *SASL_README: Postfix SASL Howto*. Postfix Documentation.

Postfix Project (n.d.f) *MILTER_README: Postfix before-queue Milter support*. Postfix Documentation. Available at: https://www.postfix.org/MILTER_README.html (Accessed: 30 April 2026).

Qiu, S., Liu, Q., Zhou, S. and Huang, W. (2022) ' Adversarial attack and defense technologies in natural language processing: a survey', *Neurocomputing*, 492, pp. 278–307. doi:10.1016/j.neucom.2022.04.020.

Qiu, S., Liu, Q., Zhou, S., Gou, M., Zeng, Y., Zhang, Z. and Wu, Z. (2025) 'Hard label

adversarial attack with high query efficiency against NLP models', *Scientific Reports*, 15, 9378. doi:10.1038/s41598-025-93566-5.

Sahami, M., Dumais, S., Heckerman, D. and Horvitz, E. (1998) ' A Bayesian approach to filtering junk e-mail', *AAAI Workshop on Learning for Text Categorization*, WS-98-05, pp. 55–62.

Sajad, U.P. (2025) ' Explainable transformer-based email phishing classification with adversarial robustness', *arXiv preprint arXiv:2511.12085*. doi:10.48550/arXiv.2511.12085.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019) ' DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter', *arXiv preprint arXiv:1910.01108*, EMC² Workshop, NeurIPS 2019.

Sbei, A., ElBedoui, K. and Barhoumi, W. (2024) ' Assessing the efficiency of transformer models with varying sizes for text classification: a study of rule-based annotation with DistilBERT and other transformers', *Vietnam Journal of Computer Science*, pp. 1–28. doi: 10.1142/S2196888824500209.

Sheffer, Y., Saint-Andre, P. and Fossati, T. (2022) *Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*. RFC 9325. RFC Editor. doi:10.17487/RFC9325.

Taha, K. (2025) 'SMART: Semantic, Multi-Objective, and Reinforcement-Based Adversarial Training for Email Spam Detection', *IEEE Access*, 13, pp. 112749–112764. doi:10.1109/ACCESS.2025.358113

Tian, Y., Dai, X., Li, Z., Guo, H. and Mao, X. (2025) ' Improving the accuracy of cybersecurity spam email detection using ensemble techniques: a stacking approach machine learning for spam email detection', *PLoS ONE*, 20(9), e0331574. doi: 10.1371/journal.pone.0331574.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. (2017) ' Attention is all you need', *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, CA, pp. 5998–6008.

Wang, C., Kuranaga, Y., Wang, Y., Zhang, M., Zheng, L., Li, X., Chen, J., Duan, H., Lin, Y. and Pan, Q. (2024) 'BREAKSPF: How Shared Infrastructures Magnify SPF Vulnerabilities Across the Internet', *Network and Distributed System Security (NDSS) Symposium 2024*, San Diego, CA, USA. doi:10.14722/ndss.2024.23113.

Wang, X. (2024) ' Spam filtering in the modern era: a review of machine learning, deep learning, and system comparisons', *2nd International Conference on Data Analysis and Machine Learning (DAML)*, pp. 451–458. SciTePress.

Bibliography

Barnes, R., Thomson, M., Pironti, A. and Langley, A. (2015) *Deprecating Secure Sockets Layer Version 3.0*. RFC 7568. RFC Editor. doi:10.17487/RFC7568.

Cormack, G.V. (2008) ' Email spam filtering: a systematic review', *Foundations and Trends in Information Retrieval*, 1(4), pp. 335–455. doi:10.1561/1500000006.

Crocker, D. (2009) *Internet Mail Architecture*. RFC 5598. RFC Editor. doi:10.17487/RFC5598.

Dent, K.D. (2003) *Postfix: The Definitive Guide*. Sebastopol, CA: O'Reilly Media.

docker-mailserver (n.d.a) *docker-mailserver: production-ready containerised mail server stack (SMTP, IMAP, LDAP, antispam, antivirus, etc.)*. GitHub repository. Available at: https://github.com/docker-mailserver/docker-mailserver (Accessed: 30 April 2026).

docker-mailserver (n.d.b) *Security: TLS (aka SSL)*. docker-mailserver documentation. Available at: https://docker-mailserver.github.io/docker-mailserver/edge/config/security/ssl/ (Accessed: 30 April 2026).

Fatima, R., Fareed, M.M.S., Ullah, S., Ahmad, G. and Mahmood, S. (2024) ' An optimized approach for detection and classification of spam emails using ensemble methods', *Wireless Personal Communications*, 139, pp. 347–373. doi: 10.1007/s11277-024-11628-9.

Gellens, R. and Klensin, J. (2011) *Message Submission for Mail*. RFC 6409. RFC Editor. doi:10.17487/RFC6409.

Géron, A. (2022) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd edn. Sebastopol, CA: O'Reilly Media.

Goodfellow, I.J., Shlens, J. and Szegedy, C. (2015) ' Explaining and harnessing adversarial examples', *3rd International Conference on Learning Representations (ICLR)*, San Diego, CA. arXiv:1412.6572.

Internet Assigned Numbers Authority (2026) *Service Name and Transport Protocol Port*

Number Registry. Available at: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml (Accessed: 16 February 2026).

Ko, A.J., LaToza, T.D. and Burnett, M.M. (2015) 'A practical guide to controlled experiments of software engineering tools with human participants', *Empirical Software Engineering*, 20(1), pp. 110–141. doi:10.1007/s10664-013-9279-3.

Lee, S., Sun, Y., Kwon, T.T., van Rijswijk-Deij, R. and Chung, T. (2022) 'Under the Hood of DANE Mismanagement in SMTP', *Proceedings of the 31st USENIX Security Symposium (USENIX Security '22)*, Boston, MA, USA.

Manning, C.D., Raghavan, P. and Schütze, H. (2008) *Introduction to Information Retrieval*. Cambridge: Cambridge University Press.

Myers, J.G. (1996) *Local Mail Transfer Protocol*. RFC 2033. RFC Editor. doi:10.17487/RFC2033.

Ng, A. (2022) *Machine Learning Specialization* [Online video lecture series]. DeepLearning.AI. Available at: https://www.youtube.com/playlist?list=PLkDaE6sCZn6FNC6YRfRQc_FbeQrF8BwGI (Accessed: 6 May 2026).

Postfix Project (n.d.a) *postscreen(8): Postfix zombie blocker*. Postfix Documentation.

Postfix Project (n.d.c) *postconf(5): Postfix configuration parameters*. Debian Manpages. Available at: https://manpages.debian.org/bookworm/postfix/postconf.5.en.html (Accessed: 16 February 2026).

Postfix Project (n.d.d) *SMTPD_ACCESS_README: SMTP relay and access control*. Postfix Documentation.

Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G.J. and Lear, E. (1996) *Address Allocation for Private Internets*. RFC 1918. RFC Editor. doi:10.17487/RFC1918.

Siemborski, J. and Melnikov, A. (2007) *SMTP Service Extension for Authentication*. RFC 4954. RFC Editor. doi:10.17487/RFC4954.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R. (2014) ' Intriguing properties of neural networks', *2nd International Conference on Learning Representations (ICLR)*, Banff, Canada. arXiv:1312.6199.

Tida, V.S. and Hsu, S. (2022) ' Universal spam detection using transfer learning of BERT model', *Proceedings of the 55th Hawaii International Conference on System Sciences (HICSS)*. doi: 10.24251/HICSS.2022.921.

Turner, S. and Polk, T. (2011) *Prohibiting Secure Sockets Layer (SSL) Version 2.0*. RFC

6176. RFC Editor. doi:10.17487/RFC6176.

Zhang, W.E., Sheng, Q.Z., Alhazmi, A. and Li, C. (2020) ' Adversarial attacks on deep-learning models in natural language processing: a survey', *ACM Transactions on Intelligent Systems and Technology*, 11(3), pp. 1–41. doi:10.1145/3374217.