Size Does Matter: The Impact of Embedding Models and Sizes on Spam Email Classification

Malte Josten[a], Gérald Kämmerer[b], Arne Kummerow[c], and Torben Weis[d]
Distributed Systems Group, University of Duisburg-Essen, Duisburg, Germany
{*malte.josten, gerald.kaemmerer, arne.kummerow, torben.weis*}*@uni-due.de*

Keywords: Email Spam Detection, Sentence Transformer, Embeddings, AI/ML

Abstract: Spam and phishing emails remain a major cybersecurity challenge, even after decades of research into reliable detection methods. Modern ML-based spam filters typically rely on text embeddings to represent email content, yet the choice of embedding model and size is often treated as secondary. This work empirically compares a diverse set of sentence embedders to assess how model type and embedding dimensionality influence downstream email spam classification. Using both classical and ML-based classifiers, we evaluate performance across multiple embedding configurations. Our results show that embedder choice - especially embedding size - substantially affects classification performance and generalisation. We observe performance differences of up to 13% overall, alongside variations of 25% in misclassified spam and 10% in misclassified ham across embedders. These findings highlight that embedding models are not interchangeable; rather, their deliberate selection is just as critical as choosing the right classifier when designing AI-based spam detection pipelines.

1 INTRODUCTION

Spam and phishing emails (hereafter collectively referred to as spam) remain a significant cybersecurity challenge, affecting both individuals and organisations on a global scale. Recent reports indicate that the volume and sophistication of such messages continue to rise (Pajola et al., 2025; Saka et al., 2025; Schäfer, 2015; Griffiths, 2025; Josten and Weis, 2025b; Kulikova, 2025). Over the years, numerous detection strategies have been proposed, ranging from rule-based and statistical models to modern machine learning (ML) and deep learning approaches. Many machine learning approaches for email processing involve converting email content into dense vector representations, often using pre-trained word or sentence embedding models. These embeddings provide the foundation for subsequent classification tasks. However, despite their central role in the detection pipeline, the choice of embedding model, including its architecture and vector dimensionality, is often treated as an afterthought. In practice, many works simply adopt a default embedder or select a model based on convenience rather than empirical justification. With this work, we demonstrate that both the classification method and the choice of embedder - including the dimensionality of the resulting embedding vector - significantly influence spam classification performance. To this end, we evaluate ten different sentence embedders, spanning a range of architectures and vector sizes (64 - 1,024), in combination with nine different classification methods, including both traditional and modern AI-based approaches. Our findings suggest that the common assumption of "the bigger, the better" in sentence embedder and dimension selection may not always hold true. Rather, it is a critical design choice that can significantly influence system robustness, particularly in the face of sophisticated spam emails. The main contributions of this paper are:

1. An email dataset with 3.45 million spam and 102k ham emails that went through rigorous preprocessing to homogenise and prepare its contents for feature extraction and binary classification.
2. A systematic evaluation of 90 embedder-classifier combinations, highlighting the critical impact of embedder choice and embedding size on spam email classification performance.

[a] https://orcid.org/0000-0003-2102-1575
[b] https://orcid.org/0009-0007-3841-7003
[c] https://orcid.org/0009-0009-2404-5530
[d] https://orcid.org/0000-0001-6594-326X

2 RELATED WORK

Traditional spam and phishing detection methods, such as blacklisting, whitelisting (Kyaw et al., 2024; Rahim and Basheer, 2021), rule-based/heuristic systems, and content-based filtering (Huang et al., 2025), form the backbone of email security but struggle against modern, dynamic attacks (Kumar et al., 2020; Blum et al., 2010; Prakash et al., 2010; S.Mahalakshmi et al., 2025). Blacklists and whitelists, for example, are reactive by design, failing to detect zero-day phishing sites, as attackers evade them via minor URL modifications (Kumar et al., 2020; Blum et al., 2010; Prakash et al., 2010). Rule-based systems, while effective for predefined patterns, e.g., suspicious keywords or URL mismatches, lack adaptability to evolving tactics (Huang et al., 2025; Chen et al., 2014). Even advanced content-based approaches like CANTINA, which leverages TF-IDF for high detection accuracy, show diminished efficacy in isolation against contemporary threats due to their static nature (Zhang et al., 2007). While historically important, these traditional methods demonstrate reduced efficacy when used in isolation, primarily due to their struggle to match the dynamic nature of current cyber threats (S.Mahalakshmi et al., 2025; Kyaw et al., 2024). While traditional ML algorithms like SVM and Random Forest have demonstrated high accuracy in cybersecurity (Mansoori et al., 2025), recent research highlights a shift toward transformer-based models and LLMs as the new standard. Comparative studies show that fine-tuned LLMs (e.g., GPT-4, BERT) surpass traditional ML approaches, establishing them as the foundation for next-generation spam filtering (Roumeliotis et al., 2024; Chataut et al., 2024; Jamal et al., 2024). LLMs are already being deployed defensively, such as in SpaLLM-Guard for high-accuracy SMS spam detection (Salman et al., 2025). Beyond performance, they enable advancements in explainability (e.g., ExplainableDetector achieving 99.84% precision with interpretable rationales (Uddin et al., 2024)) and human-induced bias reduction, improving fairness without sacrificing accuracy (Fuhnwi et al., 2025).

A recurring challenge is that many models and datasets presented in scientific literature are not publicly released. Numerous published works describe implementations that cannot be downloaded, nor easily inspected or replicated. Consequently, these models cannot be tested on other datasets, nor can their performance be validated. Because these literature models are inaccessible, we could not include them in our comparative evaluation. To compensate for this reproducibility gap, we implemented a representative set of commonly reported machine-learning approaches ourselves, systematically combining them with multiple embedding models to approximate the methodological diversity seen in prior work (see Sec. 4).

3 DATA

For this work, we used two main datasets, comprising both spam and ham emails. The spam data was obtained from the *SPAM Archive Dataset*[1], which contains spam messages collected between 1998 and today[2]. We restricted our analysis to emails originating from the years 1998 to 2024. This way, we train and test on a collection of spam emails covering over 25 years, thus including a variety of different types of spam emails. To ensure data consistency and usability, we applied extensive pre-processing: Each email was first assigned a unique identifier to facilitate traceability. Since this work focuses on the email content, email headers were removed, except for the *Subject* field, while the *To* and *From* fields were anonymised by replacing them with standardised placeholders (to@example.com and from@example.com). Only messages with a content type beginning with "text/", that could be successfully decoded using UTF-8 were retained. When necessary and applicable, alternative encodings were converted to UTF-8. The preprocessing pipeline further includes several cleaning steps: preserving all hyperlinks, stripping HTML tags, removing newline characters, and reducing multiple consecutive whitespace characters to a single space. Emails with empty message bodies after preprocessing were discarded. Finally, duplicate emails were removed based on the text of their body. After completing these steps, approximately 3.45 million unique spam messages remained. For the ham data, we employ the *190K Spam–Ham Email Dataset* available on Kaggle[3] which underwent the same preprocessing, with the additional steps of adding the header manually (as the original dataset does not contain any email header information).

From both sources, we randomly sampled 15,000 spam and 15,000 ham emails. Of these, 10,000 messages per class are used to construct the training data, while the remaining 5,000 messages per class are reserved for testing.

[1] https://untroubled.org/spam/

[2] As of March 2026

[3] https://www.kaggle.com/datasets/meruvulikith/190k-spam-ham-email-dataset-for-classification

4 METHODOLOGY

4.1 Choice of Sentence Transformer

We employ sentence-transformer models to generate dense vector representations (embeddings) of the emails' contents (see Tbl. 1). Their property of capturing semantic similarities makes them particularly suitable for tasks such as similarity-based spam classification, where lexical variations or paraphrasing can otherwise impede detection performance. The selection of transformer models was guided by current trends and usage indicators on the Hugging Face platform (Hugging Face, 2025) and usage in the literature, focusing on widely adopted and well-performing architectures. It was also guided by the aim to cover various embedding sizes (64 to 1,024 dimensions). While the common sentence embedder serves as a baseline investigation, we additionally include the *paranmt* sentence embedder (Wieting et al., 2019; Wieting et al., 2023) to broaden the benchmarking to less common embedders. Since this model is explicitly trained on paraphrastic data, incorporating it lets us examine whether paraphrase-aware training yields substantially different embeddings compared to a standard approach. For clarity and conciseness, we refer to sentence-transformer models simply as embedder (models) throughout the remainder of this paper.

We used all embedders with their default configurations, except for the *nomic* embedders. For those, we selected the "classification" task type, as it is specifically designed to extract relevant features and generate embeddings suitable for classification tasks.

4.2 Evaluation Metrics

In the context of spam and phishing detection, the primary objective is to maximise the correct identification of spam messages (true positives) while minimising the misclassification of legitimate messages (false positives), as the latter risks losing important user communication. Thus, a conservative approach is preferred: allowing some spam in the inbox is acceptable if it ensures reliable delivery of legitimate emails. To reflect this, we prioritise maximising both the True Positive Rate (TPR) and True Negative Rate (TNR). While we report accuracy for comparability with prior work, it fails to sufficiently capture the trade-off between TPR and TNR. Therefore, we also use the Geometric Mean (G-Mean):

$$\text{G-Mean} = \sqrt{TPR \times TNR},$$

which better represents their balance by penalising strong discrepancies. We avoid using the weighted G-Mean to prevent introducing additional weighting variables.

4.3 Classification Approaches

4.3.1 Individual Comparison

For the individual comparison approach, each incoming email embedding is compared against embeddings in the training data to assess similarity to known spam. Two standard similarity metrics are used: cosine similarity and Euclidean distance (L2).

To classify an email as either spam or ham, we determine a threshold value, denoted as τ, that optimises for G-Mean. For cosine similarity, whose values range from -1 to 1 (with values near 1 indicating high similarity and values near -1 indicating dissimilarity), we empirically evaluated thresholds across $\tau \in [0-1]$ due to the lack of a universally accepted threshold and its dependence on content and desired strictness. For the L2 distance, the threshold range was set to $\tau \in [0,2]$, corresponding approximately to the range of distances observed between normalised embedding vectors. Both optimisations use a step size of 0.01, as finer resolutions showed negligible performance differences.

During testing, each mail-under-test (MUT) is embedded (using the same embedder as for the training data) and compared to all known spam embeddings. The classifications are done as follows:

- Cosine similarity: MUT is SPAM if its highest similarity to known spam is $> \tau$.
- L2 distance: MUT is SPAM if its closest spam embedding distance is $< \tau$.

Table 1: Overview of sentence transformers used for creating the embeddings.

Abbr.	Sentence Transformer	Vector Size
lml6	all-MiniLM-L6-v2	384
lml12	paraphrase-multilingual-MiniLM-L12-v2	384
mpnet	all-mpnet-base-v2	768
mxbai	mxbai-embed-large-v1	64
nomic64	nomic-embed-text-v1.5	64
nomic128	nomic-embed-text-v1.5	128
nomic256	nomic-embed-text-v1.5	256
nomic512	nomic-embed-text-v1.5	512
nomic768	nomic-embed-text-v1.5	768
para	paranmt	1,024

4.3.2 k-Nearest neighbours

For kNN, we selected the number of neighbours $k \in [2, 10]$ that maximises the G-Mean. This kNN approach leverages both spam and ham examples to determine the most likely class of a given email based on the local structure of the embedding space.

4.3.3 k-Means

As a clustering-based strategy, we applied the k-Means algorithm, a classic unsupervised method that partitions data into k clusters by minimising intra-cluster variance. Since k-Means does not inherently determine the optimal number of clusters, we performed an extensive search over $k \in [2, 5000]$ with a fixed seed of 42 to ensure reproducibility, selecting the configuration that yielded the highest G-Mean. We applied the clustering on the known ham and spam embeddings separately to ensure, that the clusters stay "pure" and really only contain embeddings of one class.

4.3.4 Logistic Regression

During the training of logistic regression, we included three different maximum training iterations imax_iter $\in \{50, 100, 500\}$ for each embedder model. This resulted in 30 different trained logistic regression models, giving us a variety of classification results, which we averaged and summarised in Section 8, and analyse further in Section 5.

4.3.5 Multilayer Perceptron

To evaluate a simple neural baseline for spam classification, we implemented an MLP that operates directly on the embedding vectors. Because MLP performance can vary noticeably with network architecture, dropout rate, and random initialisation, we implemented various architecture permutations (see Sec. 8), and averaged results to obtain more stable metrics. We evaluated two feed-forward architectures, each parametrised by the featured embedder's vector dimensionality $\dim(e)$ (cf. Fig. 1 and 2). For both architectures, all intermediate dense layers use the ReLU activation function, while the final output layer uses a sigmoid activation to produce a binary spam/ham classification. The tested dropout rates were $d \in \{0.1, 0.2, 0.3\}$.

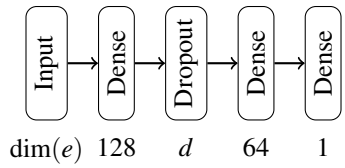


Figure 1: The first MLP architecture with two dense layers of 128 and 64 neurons, respectively, and a dropout layer in between.

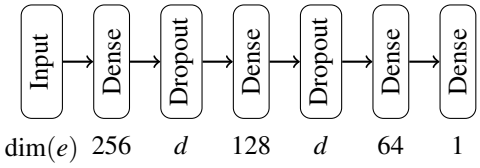


Figure 2: The second MLP architecture with an additional dense layer of 256 neurons and a dropout layer in front, followed by the two dense layers of 128 and 64 neurons, respectively, with a dropout layer in between.

4.3.6 Naive Bayes

For Naive Bayes classification, we implemented both a Gaussian and a Bernoulli classifier using their default parameter settings. Each combination of sentence embedder and Naive Bayes variant was evaluated, resulting in a total of 20 distinct configurations. Unlike other classifiers, we did not repeat runs for each configuration, as Naive Bayes produces identical results for identical input data and therefore does not exhibit variability across executions.

4.3.7 Random Forest

For the Random Forest classifier, we varied the number of trees $t \in \{80, 100, 200\}$ and evaluated each configuration under multiple initialisation seeds (see Sec. 8). This allowed us to account for stochastic effects in tree construction and to obtain more robust performance estimates across different Random Forest classifiers.

4.3.8 Support Vector Machine

To achieve meaningful SVM classification results, we implemented multiple SVMs, each using a different kernel: linear, polynomial, radial basis function, and sigmoid. This variety introduces diversity in the decision boundaries (40 different test runs) and mitigates sensitivity to a specific kernel. Performance metrics were averaged across the different kernels to provide a more reliable and representative evaluation of SVM-based classification using embedding vectors.

5 EVALUATION

In this section, we present and describe the results obtained for each classifier individually, and examine cross-classifier trends and embedder-specific behaviours, allowing us to clearly assess the impact of

the chosen embedder, and, in particular, the embedding dimensionality, on downstream spam classification performance.

5.1 Individual Comparison

Our threshold-based, individual comparison classification results reveal several important insights into the behaviour of similarity-driven spam detection when used in combination with different sentence embedders. For L2 distance, the optimal thresholds range from 0.52 to 1.00, while cosine similarity thresholds span 0.53 to 0.87. This wide dispersion demonstrates that no universal distance/similarity thresholds exists for neither L2 distance nor cosine similarity. Instead, the appropriate threshold is highly embedder-dependent, underscoring that practitioners should not rely on an arbitrary or literature-inspired constant but must validate or tune thresholds for the specific embedder they intend to deploy. Alternatively, they must reuse threshold values derived from exactly the same (or at least closely related) embedding models. Interestingly, the thresholds of some models, particularly *mxbai* and *para*, roughly overlap with the commonly cited similarity ranges in existing research (typically around 0.7-0.8) (Sandulescu and Ester, 2015; Abdelrahim et al., 2013; Josten and Weis, 2025a). However, several other embedders require substantially lower or higher thresholds to achieve optimal performance, further emphasising that generalised threshold heuristics are insufficient for robust spam classification. Second, these findings implicitly highlight the necessity of an educated and deliberate selection of the embedding model itself. The performance differences are substantial: while many embedders achieve G-Means near the 90% range, two models form notable outliers: the *para* embedder, underperforming with under 80% G-Mean, and *mxbai*, reaching only about 82%. Consequently, we observed a G-Mean spread of 12.31% and given that all models are evaluated using the same procedure and on the same dataset, such discrepancies reflect intrinsic differences in semantic representation quality, dimensionality, and model architecture, rather than differences in downstream processing. Thus, the embedder is not merely a preprocessing component but a core determinant of classification success.

Table 2 summarises the best- and worst-performing embedders for each classifier (in bold) and lists the corresponding threshold that produced the reported accuracy and G-Mean. These values represent the empirically determined optimal thresholds for each embedder in this classification setup. Overall, the results demonstrate that both embedding choice and threshold selection are pivotal elements in similarity-based spam detection. Embedders differ not only in final performance but also in the decision boundary geometry they induce, as reflected in the variety of optimal thresholds. Complete threshold-optimisation curves and extended performance visualisations for all tested embedders are provided in Section 8.

5.2 k-Nearest Neighbours

Since all embedding vectors are normalised, Euclidean distance and cosine similarity are strictly monotonically related. Consequently, we report re-

Table 2: Classification performance of cosine similarity and L2 distance, along with the optimal threshold τ for each method, evaluated across all embedder models. The best and worst performances are highlighted in bold.

	Cosine Similarity				Euclidean (L2) Distance			
Model	**TPR**	**TNR**	**Acc. (τ)**	**G-Mean (τ)**	**TPR**	**TNR**	**Acc. (τ)**	**G-Mean (τ)**
lml6	87.22	94.18	90.70 (0.53)	90.63 (0.53)	91.46	87.22	89.34 (1.00)	89.31 (1.00)
lml12	84.72	97.68	91.20 (0.60)	90.97 (0.60)	85.54	97.32	**91.43 (0.90)**	**91.24 (0.90)**
mpnet	81.20	91.16	86.18 (0.63)	86.04 (0.63)	79.82	92.58	86.20 (0.85)	86.04 (0.86)
mxbai	74.18	91.56	82.87 (0.80)	82.41 (0.80)	78.70	85.68	82.19 (0.65)	82.12 (0.65)
nomic64	80.16	95.78	87.97 (0.87)	87.82 (0.86)	82.64	93.72	88.18 (0.52)	88.01 (0.52)
nomic128	85.24	94.68	89.96 (0.82)	89.84 (0.82)	87.42	92.68	90.05 (0.61)	90.01 (0.61)
nomic256	88.54	93.86	91.20 (0.79)	91.16 (0.79)	86.92	95.60	91.26 (0.64)	91.16 (0.65)
nomic512	90.00	93.18	**91.59 (0.77)**	91.58 (0.77)	93.44	85.86	89.65 (0.70)	89.57 (0.70)
nomic768	89.88	94.46	92.17 (0.77)	**92.14 (0.77)**	93.16	88.50	90.83 (0.70)	90.80 (0.70)
para	65.46	93.10	**79.28 (0.70)**	**78.07 (0.70)**	68.24	90.34	**79.29 (0.80)**	**78.52 (0.80)**

sults using cosine similarity only, as both measures yield (near) identical neighbourhood rankings. Across all embedders, the best overall performance was consistently achieved with $k = 3$ (see Fig. 3). The resulting G-Means range from 96.81% (*mxbai*) to 98.84% (*nomic768*), corresponding to a modest performance spread of 2.04%. The largest min–max performance gap occurs at $k = 2$, where G-Means range from 96.44% (*mxbai*) to 98.67% (*nomic768*), yielding a difference of 2.33%. Overall, we observe a stable trend: *mxbai* systematically produces the lowest kNN performance, while *nomic768* achieves the highest scores across nearly all k values.

5.3 k-Means

Figure 4 presents the k-Means classification performance across all embedders and cluster counts k. In general, performance improves with increasing k and begins to stabilise beyond $k = 200$, yielding largely consistent results for values between 200 and 5,000. For the *mxbai* and *mpnet* embedders, we observe a continuous rise in G-Mean as k increases, reaching their respective maxima at $k = 5{,}000$ with 97.01% and 99.19%, respectively. Although these high cluster counts produce the best results, the performance difference between $k = 100$ and $k = 5{,}000$ is relatively small for both models. Therefore, to ensure a fair comparison with other classification approaches, and to reduce computational overhead, we report all k-Means results using $k = 100$. At $k = 100$, the lowest performing embedder is *mxbai* with a G-Mean of 95.72%, while *para* achieves the highest score at 99.30%. This corresponds to a maximum cross-embedder difference of 3.58 percentage points. For smaller cluster counts, variability is higher: for instance, at $k = 4$, G-Mean values range from 90.41% for *nomic64* to 98.15% for *para*, a spread of 7.74%.

5.4 Logistic Regression

Figure 5 illustrates the average performance of the logistic regression models. Because logistic regression is deterministic under our configuration, i.e. using identical data, no stochastic components, and only differ in the number of maximum training iterations, each run yields identical results. Consult Section 8 for a detailed depiction of further performance metrics. However, across the embedders, we observe substantial performance variation.

The weakest-performing model, *nomic64*, reaches an G-Mean of 92.87%, whereas the strongest embedder, *para*, achieves 99.19%, yielding a performance spread of 6.32 percentage points. This difference

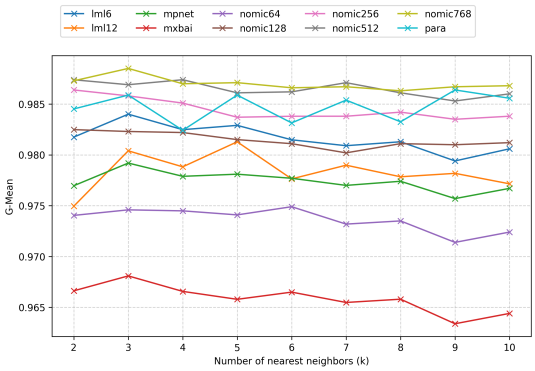
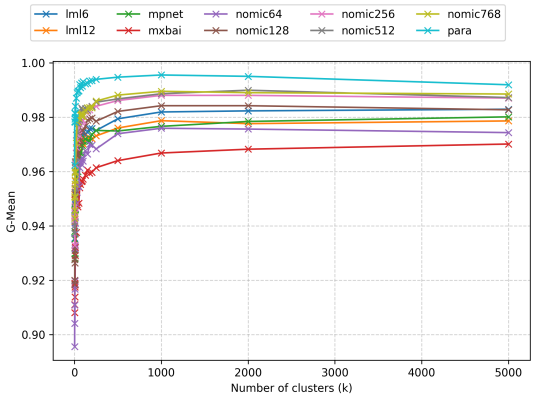
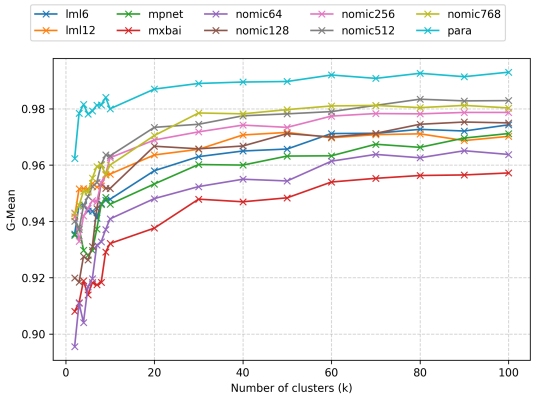


Figure 3: kNN classification performance using different embedder models and ks.



(a) Classification performance of k-Means with $2 < k < 5{,}000$.



(b) Classification performance of k-Means with $2 < k < 100$.

Figure 4: Classification performance of k-Means.

highlights once again that embedder choice is a dominant factor in downstream classification quality. Notably, the embedders follow a familiar trend observed in other classifiers: larger or semantically richer embedding models tend to outperform smaller ones, but not strictly in proportion to their dimensionality. For example, the 1,024-dimensional *para* model signif-

icantly outperforms *mpnet*, while several mid-sized nomic variants match or surpass higher-dimensional counterparts (such as *mpnet*). This suggests that also embedding quality, not dimensionality alone, drives logistic regression performance.

5.5 Multilayer Perceptron

Across all embedders, MLP performance ranges from 93.97% (*nomic64*) to 99.47% (*para*). Since the training of MLPs is more probabilistic, Figure 6 depicts not only the average, but also minimal, maximum performance, as well as its standard deviation throughout all runs. A clear trend emerges: larger embedding dimensions generally yield better average performance (see Fig. 6). For example, certain *nomic64* configurations outperform the weakest *nomic128* runs, and *mpnet* - despite its 768-dimensional vectors - performs only on par with *nomic256* and below *nomic512*. Another observation is that performance variance consistently decreases as embedding dimensionality increases. Lower-dimensional embedders exhibit relatively large fluctuations across seeds and architectures, while higher-dimensional models tend to produce more stable results.

5.6 Naive Bayes

Because Gaussian and Bernoulli Naive Bayes are deterministic, repeated runs on identical data yield constant results. Consequently, Figure 7 displays only average performances, as standard deviations and range values remain identical. Section 8 presents the resulting TPR, TNR, accuracy, and G-Mean for each embedder in more detail. Across embedders, Naive Bayes exhibits substantial performance variation. The weakest performance is observed with *nomic64* (89.24%), whereas the strongest results are obtained using *nomic768* (94.53%), leading to a spread of 5.29 percentage points in G-Mean. Surprisingly, *lml12* performs noticeably better than its equally sized counterpart *lml6*: a deviation from the trend observed in the other classifiers, where both embedders typically behaved similarly. Moreover, the best-performing embedder is not the one with the largest dimensionality. The highest-dimensional model, *para*, performs more than 3 percentage points below the top performer, demonstrating that Naive Bayes does not consistently benefit from higher-dimensional embeddings. Instead, its effectiveness appears to depend more on how well the embedding model captures separable distributions for ham and spam, an assumption closely aligned with Naive Bayes' underlying independence and Gaussian/Bernoulli likelihood models.

5.7 Random Forest

Across all embedding models and hyperparameter configurations, Random Forest classification exhibits a comparatively narrow performance range. The minimum G-Mean is achieved with the *mxbai* embeddings at 94.24%, while the *para* embeddings yield the highest G-Mean of 98.80%, resulting in a total spread of 4.56 percentage points. Overall, the results follow a trend similar to the MLP experiments (see Fig. 6), where higher-dimensional embedding models

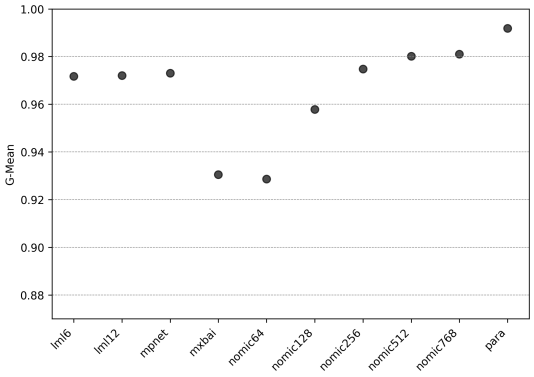


Figure 5: Logistic regression classification performance.

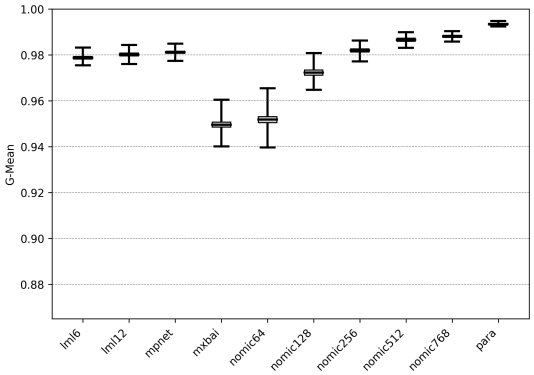


Figure 6: MLP classification performance.

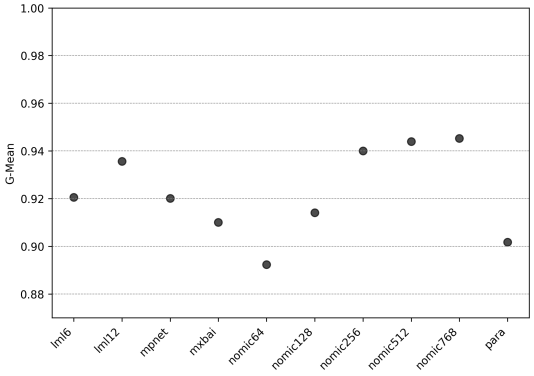


Figure 7: Naive Bayes classification performance.

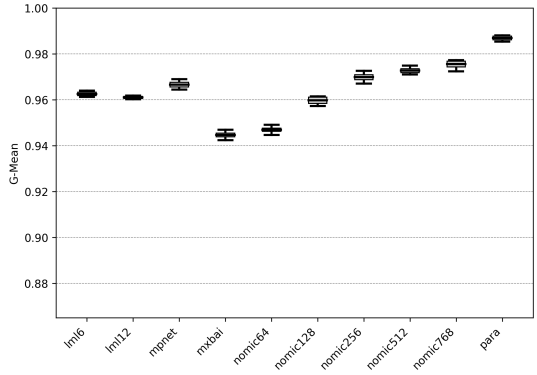


Figure 8: Random Forest classification performance.

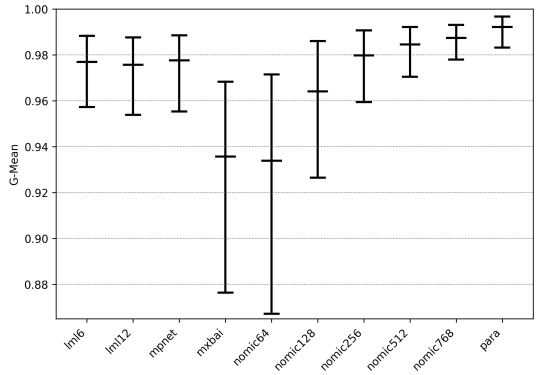


Figure 9: SVM classification performance.

tend to perform better on average. However, compared to the MLP, the Random Forest classifier shows substantially lower variance across seeds and configurations. Performance differences between embedding models are also less pronounced, leading to a more compact clustering of results and indicating that Random Forests are generally more robust to embedding dimensionality than MLPs in this context.

5.8 SVM

Since we use only non-linear kernels and set `probability=False`, the training is fully deterministic on identical input data, i.e., there is no variance when training an SVM (with the same kernel) on the same data for multiple times. Similar to logistic regression or Naive Bayes, we visualise the results in Figure 9, and give a more detailed overview of TPR and TNR in Section 8.

Across embedders, SVM performance in terms of G-Mean is consistently high, yet the results still show meaningful variation. The weakest embedder, *nomic64*, achieves a G-Mean of 93.38%, whereas the strongest, *para*, reaches 99.21%, yielding a performance spread of 5.83%. Although this variation is modest compared to the much larger spreads observed in individual classification or Naive Bayes, it nevertheless demonstrates that embedder choice substantively influences the separability of ham and spam in the induced feature space. Again, a clear trend emerges: higher-capacity embedders lead to systematically higher G-Means. Models such as *nomic512*, *nomic768*, and *para* produce embedding spaces in which the SVM can form nearly perfect decision boundaries, with G-Means above 98%. Meanwhile, lower-dimensional or noisier representations (e.g. *nomic64*, *mxbai*) result in slightly less balanced TPRs and TNRs, reflected in lower G-Mean scores. In some cases, these models surpass the high-dimensional models in some cases by up to 4.5% (*nomic64* vs. *nomic128*). This indicates, that the SVM kernels also have a crucial impact on the performance when using lower-dimensional embeddings, being the main cause for the extensive spread.

Overall, although SVMs exhibit robust performance across all embedders, the 5.83% spread in G-Mean highlights that the classifier still benefits meaningfully from stronger embedding models; an important consideration when optimising end-to-end spam detection pipelines.

6 DISCUSSION

The naive classifiers (cosine similarity and L2 distance) demonstrate reasonably strong performance across most embedders, with one notable exception: *para*. We hypothesise that its specialised training renders simple distance metrics insufficient for capturing similarity between texts that are semantically but not paraphrastically related. To investigate this further, we analysed the pairwise distances/similarities between all embeddings, which exhibit close to normal distribution (see Fig. 10). We found plateaus at the histogram's sides, suggesting that *para*'s embedding space possesses a distinct spatial structure compared to other embedders, potentially explaining its unexpectedly poor classification performance.

Our results reveal that no single classifier universally outperforms others regardless of the embedding model or dimensionality (see Fig. 11). Instead, we observe a clear trend: clustering-based algorithms, e.g. kNN, excel with smaller embeddings ($\leq$512 dimensions), while other ML-based classifiers, like SVM or MLP, gain an advantage as dimensionality increases ($>$512 dimensions). This shift underscores the importance of tailoring the embedder-classifier combination to achieve optimal performance. For instance, kNN outperforms SVM at 64 dimensions, whereas the op-

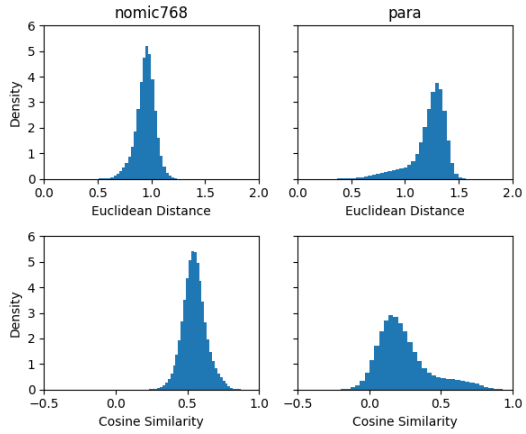


Figure 10: Comparison between the relative frequency of the pairwise distances/similarities between of all embeddings. Except the *para* histogram, every other shows a clear normal distribution (*nomic768* is used as an example).

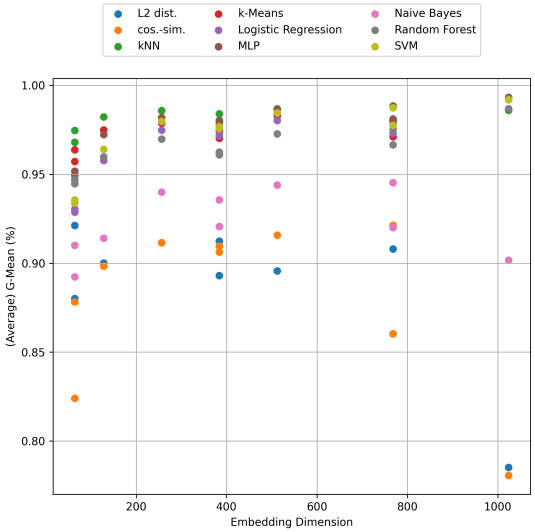


Figure 11: Summary of how the different embedding dimensions affect the (average) classification performance.

posite holds true at 768 dimensions (see Fig. 11). Thus, the interplay between embedding size and classifier choice emerges as a critical factor in maximising classification accuracy.

6.1 Limitations

The dataset, while robust for evaluating embedding-based spam classification, has key limitations. First, email entropy and structural quality vary widely, with real-world spam often being noisy, malformed, or obfuscated; factors that impact embedding quality and classifier behaviour. Though preprocessing mitigates extreme cases, residual noise and stylistic differences persist. Second, multilingual capabilities of embedding models differ, and mismatches between training data and email language/style may reduce performance comparability. Despite filtering for English, linguistic homogeneity can not be guaranteed, limiting generalisation to multilingual or code-switched environments. Third, spam is context-dependent: a message may be spam for one user but legitimate for another, so dataset labels may not reflect all real-world scenarios. Fourth, zero-day attacks (new spam variants) may evade detection, reducing performance. While embeddings capture semantic information - helping mitigate some classification performance issues - they do not ensure reliable detection of unknown spam variants. Additional limitations arise from the embedding model and classifier selection. Higher-dimensional embeddings often perform better but increase storage, memory, and inference time: factors that may be infeasible for production systems or large-scale deployments. Our work focuses on widely used, fundamental classifiers (e.g., kNN, SVM, Logistic Regression, Naive Bayes), providing a solid baseline. However, more advanced models (e.g., gradient boosting, deep learning) may offer improvements and warrant future exploration. The embedding model selection may also introduce bias. While we evaluated a representative set of contemporary sentence-transformer architectures, the broader space of models is vast and rapidly evolving. Assessing alternative architectures (e.g., domain-specific transformers) could reveal new trends or outperform current results. Finally, the embedder–classifier combinations tested here are limited. Given the substantial performance variations across embedders, expanding the search space through hyperparameter tuning, hybrid models, or ensemble methods, could provide deeper insights into the interplay between representation quality and classification strategies.

7 CONCLUSION

This work systematically evaluated diverse sentence-embedding models and classical ML classifiers for email spam detection. Results revealed a clear trend: larger embedding dimensions generally improve classification performance, while smaller models underperform regardless of classifier (cf. Fig. 11). Among embedders, *nomic768* delivered the most stable and consistently high performance across classifiers, whereas *mxbai* and *nomic64* ranked lowest. A notable outlier was *para*: it performed poorly with similarity-based classifiers and only moderately with kNN, yet achieved best-in-class results with most other classifiers. This suggests *para*'s embedding

Table 3: The summary of recorded minimum, maximum, difference, and average G-Mean performances across all embedding models.

Classifier	G-Mean Minimum	G-Mean Maximum	G-Mean Min.-Max.-Δ	G-Mean Average
cos.-sim. (+ τ)	78.07 (*para*)	92.14 (*nomic768*)	14.07	80.07
L2 dist. (+ τ)	78.52 (*para*)	91.24 (*lml12*)	12.72	87.68
kNN ($k = 2$)	96.44 (*mxbai*)	98.67 (*nomic768*)	2.33	98.02
k-Means ($k = 4$)	90.41 (*mxbai*)	98.15 (*para*)	7.74	93.98
Logistic Regression	92.87 (*nomic64*)	99.19 (*para*)	6.32	96.62
MLP	93.97 (*nomic64*)	99.47 (*para*)	5.50	97.63
Naive Bayes	89.24 (*nomic64*)	94.53 (*nomic768*)	5.29	92.24
Random Forest	94.24 (*mxbai*)	98.80 (*para*)	4.56	96.46
SVM	86.72 (*nomic64*)	99.67 (*para*)	12.95	97.08

space is ill-suited for direct distance-based discrimination but excels when paired with more expressive decision boundaries. The importance of selecting an appropriate embedding dimension is further underscored by the substantial performance differences observed across embedders. Depending on the classifier, the gap between the weakest and strongest embedder ranges from 2.33% (kNN) up to 12.95% (SVM) in G-Mean (see Tbl. 3), and the misclassification of legitimate and spam emails is as high as 11.64% (*nomic512* vs. *lml12* using L2 distance) and 25.2% (*para* vs. *nomic512* using L2 distance), respectively. Figure 11 reveals a practical trade-off: for most classifiers, doubling or tripling embedding size yields only marginal performance gains, which may not justify the increased storage and computational costs. Practitioners must weight whether slight accuracy reductions are acceptable to optimise efficiency in deployment. In a real-world context, the classification variations have profound implications: with ∼375 billion emails sent daily (The Radicati Group, 2024) and ∼50% being spam (Schäfer, 2015; Griffiths, 2025; Kulikova, 2025), a suboptimal embedder could allow up to 24.3 billion additional spam emails to reach users daily. This underscores the critical importance of rigorous model selection: embedding models are not interchangeable, and poor choices have real-world consequences. Overall, our findings demonstrate that embedding choice is at least as critical as classifier choice in modern spam-detection pipelines, and that "the bigger, the better" is not always applicable for embedding size selection.

8 DATA AVAILABILITY AND SUPPLEMENTARY MATERIAL

The source code and data of this work is publicly available at https://github.com/MalteJosten/embedding-size-spam-classification. Where applicable, we fixed the following random seeds: k-Means: 42; RF: 84, 319, 686, 823, 950. These seeds were used consistently across configurations and experimental runs to provide stable and reproducible results. All experiments were executed on 2x AMD EPYC 7252 8C 3.1 GHz, equipped with 4x NVIDIA Quadro RTX 8000 graphics cards. Depending on the embedder, embedding dimensions and classifier, the process took between a few minutes and hours. The supplementary material can be found online at https://github.com/MalteJosten/secrypt2026-size_does_matter-supplementary_material.

ACKNOWLEDGEMENTS

This paper was edited for grammar and spelling using Mistral's LeChat.

REFERENCES

Abdelrahim, A. A. A., Elhadi, A. A. E., Ibrahim, H., and Elmisbah, N. (2013). Feature selection and similarity coefficient based method for email spam filtering. In *2013 INTERNATIONAL CONFERENCE ON COMPUTING, ELECTRICAL AND ELECTRONIC ENGINEERING (ICCEEE)*, pages 630–633.

Blum, A., Wardman, B., Solorio, T., and Warner, G. (2010).

Lexical feature based phishing url detection using online learning. In *Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security*, AISec '10, page 54–60, New York, NY, USA. Association for Computing Machinery.

Chataut, R., Upadhyay, A., Usman, Y., Nankya, M., and Gyawali, P. K. (2024). Spam no more: A cross-model analysis of machine learning techniques and large language model efficacies. In *2024 8th Cyber Security in Networking Conference (CSNet)*, pages 116–122.

Chen, T.-C., Stepan, T., Dick, S., and Miller, J. (2014). An anti-phishing system employing diffused information. *ACM Trans. Inf. Syst. Secur.*, 16(4).

Fuhnwi, G. S., Revelle, M., Whitaker, B., and Izurieta, C. (2025). Using large language models to mitigate human-induced bias in sms spam: An empirical approach. In *2025 IEEE 4th International Conference on AI in Cybersecurity (ICAIC)*, pages 1–7.

Griffiths, C. (2025). The latest phishing statistics (updated october 2025): Aag it support.

Huang, W., Chu, D.-T., Bai, L.-Y., Kang, W., Zhang, H.-T., Li, B., Han, Z.-M., Ge, J., and Lin, H.-F. (2025). Evomail: Self-evolving cognitive agents for adaptive spam and phishing email defense.

Hugging Face (2025). Models compatible with the sentence-transformer library. https://huggingface.co/models?library=sentence-transformers. Accessed: 2025-06-30.

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(5).

Josten, M. and Weis, T. (2025a). Investigating the Effectiveness of Bayesian Spam Filters in Detecting LLM-Modified Spam Mails. In *Digital Forensics and Cyber Crime*, pages 285–295. Springer Nature Switzerland.

Josten, M. and Weis, T. (2025b). Large Language Models as a Cyber Threat: Towards Countering LLM-based Spam Attacks. In *2025 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 606–607.

Kulikova, T. (2025). Kaspersky spam and phishing report for 2024.

Kumar, J., Santhanavijayan, A., Janet, B., Rajendran, B., and Bindhumadhava, B. (2020). Phishing website classification and detection using machine learning. In *2020 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6.

Kyaw, P. H., Gutierrez, J., and Ghobakhlou, A. (2024). A systematic review of deep learning techniques for phishing email detection. *Electronics*, 13(19).

Mansoori, F., Chauhan, K., and Kumar, S. (2025). Deep learning and ensemble methods for robust financial email phishing detection: A performance benchmark. In *2025 4th OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 5.0*, pages 1–12.

Pajola, L., Caripoti, E., Banzer, S., Pizzi, S., Conti, M., and Apruzzese, G. (2025). E-phishgen: Unlocking novel research in phishing email detection. In *Proceedings of the 2025 Workshop on Artificial Intelligence and Security (AISec '25)*.

Prakash, P., Kumar, M., Kompella, R. R., and Gupta, M. (2010). Phishnet: Predictive blacklisting to detect phishing attacks. In *2010 Proceedings IEEE INFOCOM*, pages 1–5.

Rahim, M. N. and Basheer, K. P. M. (2021). A survey on anti-phishing techniques: From conventional methods to machine learning. *Malaya Journal of Matematik*.

Roumeliotis, K. I., Tselikas, N. D., and Nasiopoulos, D. K. (2024). Next-generation spam filtering: Comparative fine-tuning of llms, nlps, and cnn models for email spam classification. *Electronics*, 13(11).

Saka, T., Vaniea, K., and Kökciyan, N. (2025). Sok: Grouping spam and phishing email threats for smarter security. *IEEE Access*, 13:54938–54959.

Salman, M., Ikram, M., Basta, N., and Kaafar, M. A. (2025). Spallm-guard: Pairing sms spam detection using open-source and commercial llms.

Sandulescu, V. and Ester, M. (2015). Detecting singleton review spammers using semantic similarity. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 971–976, New York, NY, USA. Association for Computing Machinery.

Schäfer, C. (2015). Detection of compromised email accounts used for spamming in correlation with mail user agent access activities extracted from metadata. In *2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pages 1–6.

S.Mahalakshmi, Pansy, D., and Thejashree, V. (2025). Smart email filtering against phishing attacks. In *2025 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI)*, pages 1–6.

The Radicati Group (2024). Email Statistics Report, 2024-2028.

Uddin, M. A., Islam, M. N., Maglaras, L., Janicke, H., and Sarker, I. H. (2024). Explainabledetector: Exploring transformer-based language modeling approach for sms spam detection with explainability analysis.

Wieting, J., Gimpel, K., Neubig, G., and Berg-Kirkpatrick, T. (2019). Simple and effective paraphrastic similarity from parallel translations. In *Proceedings of the Association for Computational Linguistics*.

Wieting, J., Gimpel, K., Neubig, G., and Berg-Kirkpatrick, T. (2023). Paraphrastic representations at scale.

Zhang, Y., Hong, J. I., and Cranor, L. F. (2007). Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 639–648, New York, NY, USA. Association for Computing Machinery.