Improving Email Filtering Systems: A Graph Neural Network Approach

William Arliss
Leidos
william.f.arliss@leidos.com

ABSTRACT

This paper presents a framework for improving enterprise email filtering systems using a powerful language and communication network-based model. Emails are represented as bipartite graphs made up of user nodes and message nodes. Embeddings for user nodes are taken from a hierarchically structured communication network that links users to domains and top-domains. Email graphs are embedded and classified as malicious or benign using a heterogeneous Graph Neural Network. Training this model with a label noise-tolerant loss function allows training labels to be taken from an imperfect existing filtering system, which can afford a large volume of data without the expensive requirement of clean labels.

CCS CONCEPTS

• **Computing methodologies → Machine learning approaches**; • **Security and privacy → Phishing**.

KEYWORDS

machine learning, neural networks, label noise, email filtering

1 INTRODUCTION

Enterprise email networks provide a broad setting in which a company's security may be threatened. Through email, malicious entities conduct social engineering attacks — referred to as "phishing" — to compromise data and inject malware [18]. Email threats are subtle, dangerous, and have a widespread impact; in a survey spanning seven countries, 86% of organizations reportedly experienced bulk phishing in 2021. Of those respondents, 83% reported successful attacks perpetrated against them [18].

Companies defend themselves from phishing attacks with email filtering systems: scanners that filter out unwanted emails using a combination of content-based rules, blacklists, whitelists, malware detection, and threat intelligence [4, 18]. Machine learning (ML) also provides an adaptable tool set for combating the ever-evolving landscape of email threats [4, 8]. Recent advances in the ML fields of Natural Language Processing (NLP) and Graph Neural Networks (GNNs) continue to strengthen email defenses.

Efforts to apply ML to email filtering are often hindered by the requirement of a large body of reliably labeled data [23]. Enterprises with existing filtering systems may turn to their own archives and derive labels from records of which emails were filtered. However, this means that in the optimal case of a model learning the targets perfectly, it will still miss all the same attacks that the existing system does. Such a case of unreliable ground-truth is an example of the broader issue in ML of label noise [7, 21].

KDD AI4Cyber'23, August 7, 2023, Long Beach, CA, USA
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

While many ML applications for email filtering are focused on text features, another useful feature set lies with the social interactions in an email network [1]. Examining one user's relation to another adds context to the text of an exchange and can inform filtering decisions. Even more information can be gained by considering the relation between one user and the domain name or top-domain of another.

This paper presents a modeling procedure for improving email filtering systems through two main contributions. The first contribution is the use of a holistic email representation that combines text embeddings with hierarchically structured domain and user embeddings from an email communication network. This representation affords stronger performance than its component parts. The second contribution is the application of label noise mitigation techniques for deep neural networks (DNNs) to the email filtering setting. With noise-tolerant loss functions, a model can achieve superior performance to the existing system from which its imperfect training labels are taken.

2 RELATED WORK

2.0.1 ML and Emails. Applying ML to email filtering has been in practice for many years and is surveyed in [4, 8]. One of the earliest applications is [19], in which emails are prepared in a sparse bag-of-words representation and classified with the Naive-Bayes algorithm. In [30], the adaptability of TF-IDF vectorization and Naive-Bayes to label-flipping attacks on training data is studied.

The Naive-Bayes classifier is compared to a Multi-Layer Perceptron (MLP) neural network for spam filtering in [22]. In [14], a DistilBERT [20] model is fine-tuned with content (text-based) and context (email header-based) features to determine the maliciousness of emails.

Statistical social network features and language embeddings are used in [1] to attribute nodes in a bipartite graph of emails and a GNN is used to classify emails as work-related or personal. A Semantic GNN is adopted in [16] to learn linguistic features of malicious emails and improve filtering performance.

2.0.2 Label Noise. The problem of supervised learning in the presence of label noise is also a topic of much discussion in ML research [6, 7, 23, 31]. Label noise occurs when the targets used for training have been corrupted from the ground-truth [23]. The corrupted label set is referred to as the "silver" labels and the ground-truth labels are referred to as "golden" labels.

It has been shown that models trained on silver labels generalize poorly in testing [7, 13, 23, 31]. Experimental evaluations in [3, 21, 30] demonstrate that classical ML-based email filtering techniques underperform when trained on unreliable targets. Both [3] and [21] specifically observe that filtering models trained on labels sourced from imperfect third parties incur serious performance degradation due to over-fitting. This paper extends these observations to DNN-based email filtering techniques.

Performance degradation caused by over-fitting to noisy labels is a particular problem for DNNs, as shown in [2, 28, 29]. Robustness to this degradation is examined in [9] and conditions for noise-tolerant loss functions for DNNs are proposed. The work on noise-tolerance is extended in [15] by introducing a general set of robust loss functions that can address uniform noise. Other loss functions for addressing label noise are proposed in [5, 10, 17, 24, 27, 31].

A limitation of [1] is that emails are classified within the context of a full graph. This is incompatible with the email filtering setting, in which emails must be classified quickly and individually as they arrive. A limitation of [3, 21, 30] is that robustness to label noise is not addressed in the context of deep learning. This paper addresses the limitations of both works by generalizing offline node classification to online subgraph classification for email filtering and by training against a noise-tolerant objective.

3 THEORETICAL APPROACH

3.1 Email Graph Attribution

Emails take the form of attributed graphs of the interaction between two or more user-entities and the text of the exchange [1]. Transforming this data structure into a numeric representation is challenging because the set of natural language and user-entities must be encoded into the same vector space. The following sections describe a graph-based approach for representing email text and user interactions holistically.

3.1.1 Email-Based Communication Networks. The full set of email communications between unique users in a dataset is captured in a *user-message network*[1] [1]. The bipartite *user-message network* $\mathcal{B} = (U, M, E)$ consists of two sets of nodes: users and messages. User nodes represent user-entities (e.g., individual people, notification generators, help desks, and front offices). Message nodes represent the text of an email communication between two user-entities. Edges between the two sets of nodes indicate which users are sending or receiving which messages. A hypothetical *user-message network* is shown in figure 1.a.

The *user-message network* is constructed from subgraphs derived from individual email correspondences. Each *user-message subgraph* $\mathcal{B}_s \subset \mathcal{B}$ consists of a user node representing the sender of the email, a message node representing the text (subject and body), and at least one user node representing the recipient(s) (addresses in the "to", "cc", and "bcc" header fields). Directed edges from the sender to the message and from the message to each recipient specify the role of each user in the exchange. In the example network of figure 1.a, the subgraph $\mathcal{B}_s = (\{U_1, U_2\}, \{M_1\}, E_s)$ represents a single email correspondence from U_1 to U_2.

The *user network* $\mathcal{G}_U = (U, E)$ is a projection of the *user-message network* to a graph consisting only of user nodes [1]. A single edge exists between users U_i and U_j in this network if a path (U_i, M_k, U_j) exists in the *user-message network*. Edges are weighted by the number of these paths that exist — i.e., the number of times user U_i has emailed user U_j.

[1]This is referred to in [1] as the "*bipartite email-user network*".

The *user network* can be extended by noting an interesting property of email addresses. When determining the validity of the sender of a suspicious email, it is intuitive to consider individual components of an email address instead of simply the unique entity. If the full email address of the sender is unfamiliar to the recipient, a logical next step would be to consider the familiarity of the domain name in the address. Going a step further, if the domain name is unfamiliar, one might examine the top-domain.

Email addresses have a hierarchical structure consisting of full addresses at the bottom, domain names in the middle, and top-domains at the top. Full addresses correspond to the user nodes in the *user-message network*, denoted U. Domain names make up the set of nodes denoted D, where $D \supset U$. Top-domains make up the set of nodes denoted T, where $T \supset D \supset U$.

This hierarchical property motivates the augmentation of the *user network* to the *extended user network* $\mathcal{G} = (U, D, T, E)$. This extended network consists of the three node sets and nine connected edge blocks. Each block is made up of the (weighted) connections from one node set to another. Naturally, the block of connections from user nodes to user nodes, denoted $\mathcal{G}_{UU}$, is simply the original *user network* $\mathcal{G}_U$.

The block of connections within the domain node subset, denoted $\mathcal{G}_{DD}$, represents the aggregation of email communications between domain names. Here, an edge exists between domains $D_{i'}$ and $D_{j'}$ if there is an edge between any child nodes $U_i \in D_{i'}$ and $U_j \in D_{j'}$. The block of connections within the top-domain node subset, denoted $\mathcal{G}_{TT}$, represents the aggregation of communications between top-domain nodes. Here, edges exist between top-domains $T_{i''}$ and $T_{j''}$ if there is an edge between any grandchild nodes $U_i \in D_{i'} \in T_{i''}$ and $U_j \in D_{j'} \in T_{j''}$. The remaining six blocks represent aggregations between distinct node subsets. Discussion of the *extended user network* is continued in appendix C.

3.1.2 Message Features. Following [1], different types of features are extracted for different types of nodes in the *user-message network*: text features for message nodes and network features for user nodes.

Extending the use of text embeddings in [1], this paper uses a pre-trained MiniLM[2] sentence transformer based on [26]. The model is fine-tuned on an email corpus following the Transformer-based Sequential Denoising Auto-Encoder (TSDAE) procedure from [25]. The resulting message embedding model is referred to as MsgAttr (equation 1).

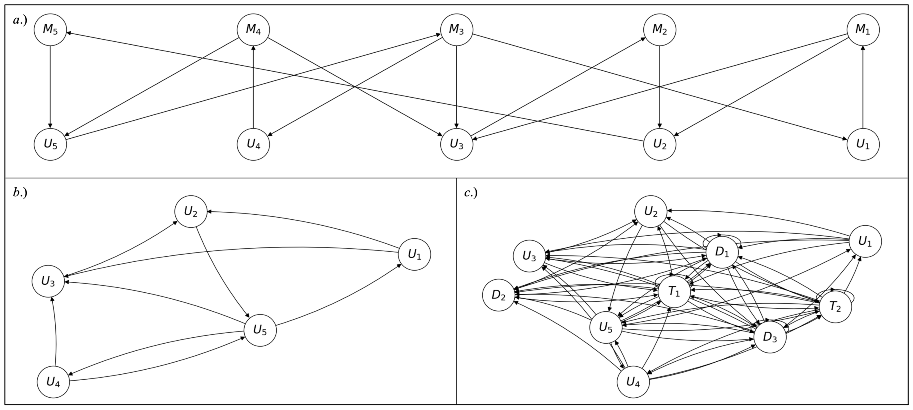
$$\text{MsgAttr}\colon M_i \mapsto \mathbb{R}^{384} \tag{1}$$

Documents in the email corpus consist of subject lines and message bodies extracted from emails. The documents are parsed to remove non-informative punctuation, UTF codes, and strings of more than 15 characters (this filters out noisy alphanumeric strings appended to many email bodies). URLs, IP address, domain names, and timestamps are all replaced with identifying tokens. All alphabetic characters are standardized to lowercase.

3.1.3 User Features. The *extended user network* offers a rich source of social network information to be used for user node features. One particularly useful set of information that can be extracted is node embeddings. The GraphSAGE algorithm [11] is used to extract

[2]MiniLM: https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2.

Figure 1: Email-based communication network representations



a.) A bipartite *user-message network* comprised of five users and five messages. *b.*) The homogeneous *user network* projection of the graph in *a*. *c.*) The *extended user network* incorporating the set of domains and top-domains. *(Source: author's calculations.)*

embeddings from the *extended user network* to attribute user nodes in the *user-message network*.

GraphSAGE generates an embedding for a given node by updating the node's own latent feature vector with an aggregation of its neighbors' features. The authors of [11] show that using the max-pooling neighborhood aggregation function allows for structural learning on nodes initialized with arbitrary features. Accordingly, this paper initializes feature vectors for each node with random draws from a normal distribution. The full embedding model is made up of multiple GraphSAGE layers stacked together and is trained with the link prediction objective.

Using GraphSAGE this way, arbitrary features given to each node in the *extended user network* can be mapped to information-rich embeddings for downstream modeling. Embeddings are extracted for all nodes $U \cup D \cup T$, meaning that a given user-entity in the *user-message network* will have three associated feature vectors: one for their full username, one for their domain name, and one for their top-domain. The full function for user embeddings is given by

$$\mathrm{UsrAttr}(U_i) = \frac{|U|\, h(U_i) + |D|\, h(D_{i'}) + |T|\, h(T_{i''})}{|U| + |D| + |T|} \tag{2}$$

where h is the function for extracting vectors from the *extended user network* using the full GraphSAGE embedding model.

Taking a weighted average of these three vectors yields better performance in downstream modeling than any individual vector alone. This is likely due to the size and sparsity of the *user network*. Enterprise email networks contain many users, including internal users who never communicate with one another and external users who only appear once or twice. Incorporating the domain and top-domain blocks of $\mathcal{G}$ in analysis — effectively examining relationships between domains and top-domains instead of just between users — affords much more informative features than the sparse user block alone.

In summary, the *user-message network* $\mathcal{B}$ is made up of individual email communications represented as *user-message subgraphs* $\mathcal{B}_s$. Each *user-message subgraph* consists of user nodes and message nodes. The MsgAttr function (equation 1) is used to attribute message nodes in $\{\mathcal{B}_s\}$ and the UsrAttr function (equation 2) is used to attribute user nodes. It should be noted that this paper differs from the approach of [1] in that the GraphSAGE embedding task is separated from the classification task. Doing so changes the modeling process from one of node classification in a full *user-message graph* to one of graph classification in a set of *user-message subgraphs*.

3.2 Modeling Under Label Noise

3.2.1 Architecture. The email filtering model discussed here is made up of a GNN encoder module, which creates vector embeddings of *user-message subgraphs*, followed by an MLP module, which estimates probabilities of each input being malicious or benign.

The graph encoder module has the job of transforming the set of input graphs $\{\mathcal{B}_s\}$ into latent vector representations X. This paper uses the bipartite extension of graph convolution [12] from [1] to accommodate heterogeneous graphs, where different weight matrices encode the features of different node types.

Next, a pooling layer is used to flatten the encoded subgraphs into dense vectors to be used by downstream layers. This is done by concatenating the latent vector of the message node with the average feature vector of the user nodes in the subgraph. This embedding is then passed through a series of dense layers to estimate the probability distribution p of the label y given the input x.

$$\mathrm{CE}(p, y) = -\textstyle\sum_{i=1}^{k} q_i \ln (p_i) \tag{3}$$

The model is trained using cross-entropy loss as given in equation 3, where k is the number of classes in the label set (two in the case of email filtering) and q is a one-hot encoded vector such that $q_y = 1$.

3.2.2 Label Noise Robustness. The assumption of label noise is used to address the effects of sourcing training labels from an existing email filtering system. The risk that a trained model makes the same mistakes as the existing system is mitigated by adopting a label noise-tolerant loss function.

It is shown in [9] that label noise can be addressed under certain conditions. One main requirement is that the loss function $\mathcal{L}$ used in training is symmetric, satisfying $\sum_{i=1}^{k} \mathcal{L}\left(f(x), i\right) = C$ for all samples $x \in X$ and for any model f, where C is a constant.

This condition is met for most classification loss functions with the incorporation of a normalizing term [15]. A set of normalized loss functions is proposed in [15] under a loss framework called

Active-Passive Loss (APL). Normalized loss functions take the form $\bar{\mathcal{L}} = \frac{\mathcal{L}(f(x),y)}{\sum_{i=1}^{k} \mathcal{L}(f(x),i)}$ and are bounded between 0 and 1. The authors suggest combining two such functions: one that behaves "actively" — only maximizing the predicted probability of the target class — and one that behaves "passively" — also minimizing predicted probabilities of the off-target classes.

$$\text{APL} = \alpha \, \text{NCE}(p, y) + \beta \, \text{MAE}(p, y) \tag{4}$$

$$\text{NCE} = \frac{\sum_{i=1}^{k} q_i \ln(p_i)}{\sum_{i=1}^{k} \sum_{j=1}^{k} q_i \ln(p_j)} \tag{5}$$

$$\text{MAE} = \sum_{i=1}^{k} |q_i - y_i| \tag{6}$$

Effective learning under label noise is demonstrated in [15] by combining normalized cross-entropy (equation 5) for the active portion and mean absolute error (equation 6) for the passive portion. The combined loss function is shown in equation 4, where α and β are hyper-parameters controlling the mixture strength of the active and passive components.

4 EXPERIMENTS

4.1 Data

To test the effectiveness of this email filtering approach, a dataset that fulfills certain requirements is needed. First, because the feature extraction process in section 3.1 relies heavily on social network characteristics, the email corpus used for training and testing must exhibit some social network structure. The dataset must also come with reliable ground-truth to allow for the simulation of label noise.

This paper uses the combined TREC-06, TREC-07, and TREC-08 corpora from the Text REtrieval Conference[3]. Many emails across the three corpora are centered around users from the University of Waterloo (where the dataset is maintained) or users from a related educational institution. A portion of the data also overlaps with the Enron[4] dataset. Note that these corpora are used in [3, 16, 21, 22, 30].

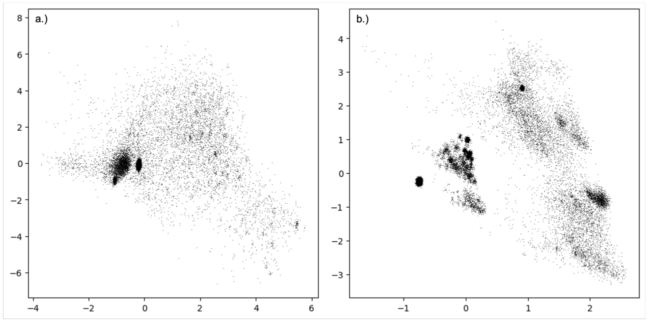
Prior to experimentation, a number of emails are heuristically removed: duplicate emails and emails for which the sender or recipient(s) is undisclosed, has been redacted, or is otherwise invalid (e.g., has no domain name). The dataset is down-sampled such that malicious emails make up only 25% of the total corpus.

The processed dataset is made up of 95,376 emails. For testing, 1,000 malicious and 1,000 benign emails are held out. There are 93,376 emails used for training, 23,344 of which are malicious and 70, 032 of which are benign. The training emails are used to build the *extended user network* (summary statistics in appendix B).

4.2 Results

4.2.1 Node Embeddings. To begin discussion of the results, consider first the embeddings from UsrAttr. Figure 2 shows a 2D projection of two embedding techniques. Panel 2.a visualizes embeddings generated by a traditional GraphSAGE model trained on the *user network* of the TREC corpus. This approach does not fully capture information on domain and top-domain interactions; consequently, there are only two or three well defined communities and many outlying nodes. Panel 2.b visualizes embeddings generated by UsrAttr trained on the *extended user network*. These embeddings reveal a clear community structure with several dense clusters.

Figure 2: User embeddings comparision



a.) Embeddings generated with standard GraphSAGE trained on the *user network*. *b.*) Embeddings generated with UsrAttr trained on the *extended user network*. *(Source: author's calculations.)*

Having more clusters is preferable in this context because it implies a richer feature space. Clusters in the node embeddings imply community structure in the network. Embeddings that fail to encode the clusters also fail to capture the community structure. Using these embeddings for downstream modeling also improves classification results.

4.2.2 Feature Sets. To demonstrate the effectiveness of combining node and text embeddings in a GNN, classification experiments are run using four different feature sets. The first model is trained only on user node embeddings from the *user network* extracted with standard GraphSAGE (figure 2.a). The second model is trained only on user node embeddings from the *extended user network* extracted with UsrAttr (equation 2, figure 2.b). The third model is trained only on message embeddings extracted with MsgAttr (equation 1). The fourth model uses the architecture outlined in section 3.2.1 and is trained on combined embeddings from UsrAttr and MsgAttr.

Table 1: Modeling results on different feature sets

Features	Recall	Precision	F1	Accuracy
GraphSAGE	0.936 ± 0.012	0.938 ± 0.001	0.937 ± 0.006	0.937 ± 0.005
UsrAttr	0.952 ± 0.008	0.956 ± 0.004	0.954 ± 0.004	0.954 ± 0.003
MsgAttr	0.968 ± 0.005	0.984 ± 0.005	0.976 ± 0.002	0.976 ± 0.002
Combined	0.980 ± 0.002	0.985 ± 0.003	0.982 ± 0.001	0.982 ± 0.001

"Combined" refers to combined UsrAttr and MsgAttr features in the GNN architecture from section 3.2.1. *(Source: author's calculations.)*

Each model uses the same MLP architecture, the same hyper-parameter configuration, and is trained on cross-entropy loss. Results against the holdout set are aggregated over five runs and shown in table 1. The user node embedding approach described in this paper achieves superior performance to standard GraphSAGE in downstream modeling. The transformer-based message embeddings are also very effective on their own. Performance is further improved by combining the message and user embeddings in a GNN, achieving 98% testing accuracy.

Although the datasets are not perfectly comparable (after the pre-processing described in section 4.1), these results are an improvement upon those in [16, 22, 30]. Using a Semantic GNN on email messages, [16] achieves 96.6% testing accuracy. Using a MLP

[3]TREC Spam Track: https://trec.nist.gov/data/spam.html.
[4]Enron Email Dataset: https://www.cs.cmu.edu/~enron/.

on vectorized text, [22] achieves 93% accuracy. Using the Naive-Bayes algorithm on TF-IDF vectors with clean labels, [30] achieves 89.9% accuracy.

4.2.3 *Noise Robustness.*

As a simple stand-in for an existing rule-based email filtering system for the TREC data, this paper fits a decision tree to the TF-IDF vectors extracted from each email's subject line and body. The predictions from this "filterer" are used as the silver label set for training a new model. The filterer's predictions match the golden training labels with 90.9% accuracy, implying a roughly 10% level of label noise. The performance of the filterer on the the holdout data is shown in the first row of table 2.

Table 2: Modeling results under label noise

Loss	Recall	Precision	F1	Accuracy
	0.771	0.849	0.808	0.863
CE	0.86 ± 0.027	0.916 ± 0.009	0.887 ± 0.014	0.89 ± 0.012
APL	0.906 ± 0.026	0.931 ± 0.009	0.918 ± 0.011	0.919 ± 0.009
NB	0.802 ± 0.0	0.956 ± 0.0	0.872 ± 0.0	0.883 ± 0.0

First row: comparison of filterer (silver labels) against held-out golden labels. Second row: performance of GNN trained with cross-entropy loss. Third row: performance of GNN trained with active-passive loss. Fourth row: performance of Naive-Bayes algorithm. *(Source: author's calculations.)*

Using the same architecture and training procedure as the combined model in table 1 affords markedly lower performance on the golden holdout set when trained on the silver labels. Surprisingly the model still outperforms the filterer using non-robust cross-entropy loss. This is likely due to the richness of the features used. When trained using active-passive loss instead, the model recovers an impressive amount of accuracy and recall.

The failure of the robust model to achieve more than 92% accuracy could be due to the type of label noise present. Using the filterer's predictions as silver labels in training introduces feature-dependent label noise [7, 21] to the modeling process. It is possible that such noise is too complicated to be addressed by symmetric loss functions alone.

In a set of experiments on the TREC-07 dataset, [30] fits the Naive-Bayes algorithm under varying label noise levels. In the presence of 10% noise, their approach achieves 78.1 – 88.8% accuracy. When applied to this paper's dataset, the same algorithm achieves 88.3% accuracy.

5 CONCLUSION

This paper demonstrates the usefulness of the *extended user network* in email filtering. Incorporating text features from MsgAttr and network features from UsrAttr in a graph classification task achieves strong performance in email filtering. Label noise is effectively addressed by training with a noise-tolerant loss function. Taken together, these two components yield a framework for improving enterprise email filtering without the need for expensive ground-truth labels.

ACKNOWLEDGMENTS

The author thanks the anonymous reviewers for their helpful comments on this paper. Special thanks is given to Dr. Graham Mueller and Dr. Alex Rosenfeld for their constructive reviews and valuable guidance throughout the project.

REFERENCES

[1] S. B. Alkhereyfy and O. Rambow. Email classification incorporating social networks and thread structure. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, Marseille, France, May 2020. European Language Resources Association.
[2] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien. A closer look at memorization in deep networks, 2017.
[3] G. V. Cormack and A. Kolcz. Spam filter evaluation with imprecise ground truth. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 604–611, New York, NY, USA, 2009. Association for Computing Machinery.
[4] E. G. Dada, J. S. Bassi, H. Chiroma, A. O. Adetunmbi, O. E. Ajibuwa, et al. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6):e01802, 2019.
[5] E. Englesson and H. Azizpour. Generalized jensen-shannon divergence loss for learning with noisy labels, 2021.
[6] B. Frénay and A. Kabán. A comprehensive introduction to label noise. In *The European Symposium on Artificial Neural Networks*, 2014.
[7] B. Frenay and M. Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014.
[8] T. Gangavarapu, C. D. Jaidhar, and B. Chanduka. Applicability of machine learning in spam and phishing email filtering: Review and approaches. *Artif. Intell. Rev.*, 53(7):5019–5081, oct 2020.
[9] A. Ghosh, H. Kumar, and P. S. Sastry. Robust loss functions under label noise for deep neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 1919–1925. AAAI Press, 2017.
[10] A. Ghosh and A. Lan. Contrastive learning improves model robustness under label noise, 2021.
[11] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017.
[12] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017.
[13] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition, 2016.
[14] Y. Lee, J. Saxe, and R. E. Harang. CATBERT: context-aware tiny BERT for detecting social engineering emails. *CoRR*, abs/2010.03484, 2020.
[15] X. Ma, H. Huang, Y. Wang, S. Romano, S. M. Erfani, and J. Bailey. Normalized loss functions for deep learning with noisy labels. *CoRR*, abs/2006.13554, 2020.
[16] W. Pan, J. Li, L. Gao, L. Yue, Y. Yang, L. Deng, C. Deng, and S. Ali. Semantic graph neural network: A conversion from spam email classification to graph classification. *Sci. Program.*, 2022, jan 2022.
[17] G. Patrini, A. Rozza, A. Menon, R. Nock, and L. Qu. Making deep neural networks robust to label noise: a loss correction approach, 2017.
[18] Proofpoint. 2022 state of the phish - an in-depth exploration of user awareness, vulnerability and resilience. Technical report, Proofpoint, Inc., 2022.
[19] J. Rennie et al. An application of machine learning to e-mail filtering. In *Proc. KDD-2000 Text Mining Workshop*, pages 75–80. Citeseer, 2000.
[20] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.
[21] D. Sculley and G. Cormack. Filtering email spam in the presence of noisy user feedback. In *International Conference on Email and Anti-Spam (CEAS)*, 2008.
[22] A. Sharma, S. Kumar, and M. Aslam. A comparative study between naive bayes and neural network (mlp) classifier for spam email detection. 2014.
[23] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee. Learning from noisy labels with deep neural networks: A survey, 2022.
[24] B. van Rooyen, A. K. Menon, and R. C. Williamson. Learning with symmetric label noise: The importance of being unhinged, 2015.
[25] K. Wang, N. Reimers, and I. Gurevych. Tsdae: Using transformer-based sequential denoising auto-encoderfor unsupervised sentence embedding learning. *arXiv preprint arXiv:2104.06979*, 4 2021.
[26] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.
[27] X. Wang, Y. Hua, E. Kodirov, and N. M. Robertson. Imae for noise-robust learning: Mean absolute error does not treat examples equally and gradient magnitude's variance matters, 2020.
[28] Q. Yao, H. Yang, B. Han, G. Niu, and J. Kwok. Searching to exploit memorization effect in learning from corrupted labels, 2020.
[29] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization, 2017.
[30] H. Zhang, N. Cheng, Y. Zhang, and Z. Li. Label flipping attacks against naive bayes on spam filtering systems. *Applied Intelligence*, 51(7):4503–4514, jul 2021.
[31] Z. Zhang and M. R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels, 2018.

A IMPLEMENTATION

The UsrAttr model uses hidden layers of size 40, 70, and 100. The model is trained for 30 epochs with an initial learning rate of 0.01 and an exponential decay rate of 0.97. The weight decay strength used is 0.0001. For link prediction, 10 negative edges are sampled for every positive edge.

The TSDAE fine-tuning procedure for the UsrAttr model is performed for 1,000 steps with batches of size 64 and a constant learning rate of 0.00005.

For the classification models in section 4.2.2 that use user embeddings alone, an aggregation function must be used to create a single vector that incorporates (directed) information about an email's sender and recipients. This paper uses

$$\bar{x} = \frac{x_u}{\|x_u\|_2} + \frac{1}{|v|} \sum_{i \in v} \frac{x_i}{\|x_u\|_2}$$

where u is the sender of the email, v is the set of recipients of the email, and x_i is the embedding of the i^{th} user node.

The MLP architecture shared by all classifiers uses hidden layers of size 240 and 120 each with batch normalization and ReLU activation functions. The output is of dimension 2 and uses a Softmax activation function. The models are trained for 30 epochs with Adam optimization using weight decay strength of 0.00005. The initial learning rate is 0.005 and the exponential decay rate is 0.95. To address imbalance in the labels, negative and positive samples are re-weighted $1 : 3.5$ in the loss function.

For the noise-tolerant training in section 4.2.3, APL (equation 4) is used with $\alpha = 0.5$ and $\beta = 0.25$. The Naive-Bayes algorithm reproduced from [30] is applied to TF-IDF vectorized $(2, 3)$-grams and is tuned using grid-search hyper-parameter optimization with a Multinomial prior.

B NETWORK STATISTICS

Table 3 shows network statistics derived from the *extended user network* built from the TREC corpus (section 4.1). The first row of the table shows statistics for the full graph $\mathcal{G}$, the second row shows statistics for just the user block $\mathcal{G}_U$, the third row shows statistics for the domain block $\mathcal{G}_D$, and the fourth row shows statistics for the top-domain block $\mathcal{G}_T$. Note that the third column shows the number of connected components. The final column measures total degree, weighted by the function ω (section 3.1.1).

Table 3: *Extended user network* statistics

	Nodes	Edges	Conn. Comp.	In-Degree		Out-Degree		ω Degree	
	count	*count*	*count*	*mean*	*median*	*mean*	*median*	*mean*	*median*
$\mathcal{G}$	54,853	241,737	3	4.4	0	4.4	3	11.3	1
$\mathcal{G}_U$	37,060	54,583	1,606	1.5	0	1.5	1	5.6	1
$\mathcal{G}_D$	17,588	27,205	347	1.6	0	1.6	1	11.8	1
$\mathcal{G}_T$	205	1,108	3	4.9	0	4.9	2	101.6	3

(Source: author's calculations.)

C FULL ADJACENCY MATRIX

The adjacency matrix A of the *extended user network* is made up of nine blocks. Each block is itself an adjacency matrix between two node sets. For example, the (weighted) adjacency matrix of the user-user block is denoted A_{UU} and its elements are $(A_{UU})_{ij} = \omega(U_i, U_j)$, where ω counts the number of emails sent between two nodes or their children. The full matrix is written as

$$A = \begin{pmatrix} A_{UU} & A_{UD} & A_{UT} \\ A_{DU} & A_{DD} & A_{DT} \\ A_{TU} & A_{TD} & A_{TT} \end{pmatrix}$$

The hierarchical relationship between the three node sets implies that each of the nine blocks has the same total weighted degree, equal to d. That is, $\sum_i \sum_j (A_{UU})_{ij} = \sum_i \sum_j (A_{UD})_{ij} = ... = d$ for all permutations of $\{U, D, T\}$, and thus $\sum_i \sum_j A_{ij} = 9d$. It is also noteworthy that $\omega(U_i, D_{j'}) = \sum_{U_j \in D_{j'}} \omega(U_i, U_j)$ and that $\omega(D_{i'}, D_{j'}) = \sum_{U_i \in D_{i'}} \sum_{U_j \in D_{j'}} \omega(U_i, U_j)$ and so on.