Inbox Guardian: A Machine Learning Framework for Customized Email Spam Detection

John Guerrerio
john.j.guerrerio.26@dartmouth.edu
Dartmouth College

Abstract

Filtering email spam from "ham" (desirable messages) is an important task to protect email users from scams and keep their inboxes clear. Machine learning offers a promising approach to perform this filtering. Methods such as Naïve Bayes, K-Nearest Neighbors, and BERT have achieved high performance on large, general-purpose spam/ham datasets. However, the effectiveness of these methods as user-specific spam filters is unknown. Here, we demonstrate a system that allows users to train their own custom spam classifiers and evaluate the effectiveness of several common machine learning methods on this classification task. We find that methods such as Naïve Bayes, Logistic Regression, and Support Vector Machines offer promising performance on this task. Naïve Bayes in particular achieved a test f1 of 0.92 with Tf-Idf features, indicating that it would perform well in a production version of this system. Our results show these machine learning methods can effectively classify emails based on individual user preferences, a low-resource hyper specific classification task. We anticipate this work to be a starting point for future development of custom email filters.

1 Introduction

Email spam is a problem for everyone, ranging from a daily annoyance to a threat to one's computer security and financial resources. Automatically classifying emails as spam or "ham" (legitimate emails the recipient would want to receive) can keep one's inbox clear and potentially prevent one from falling for a phishing scam. Mohammed, et al. presents early work in this direction; they use several non-deep learning machine learning methods such as Naïve Bayes, Support Vector Machines, and K-Nearest Neighbors to train a spam/ham classifier on the Email-1431 dataset (Mohammed, 2013). Gangavarapu et al. furthers this work via a larger dataset. They develop their own dataset of 3,844 emails, split evenly spam and ham. They then train a Naïve Bayes classifier, a Support Vector Machine, and several ensemble-based approaches (Gangavarapu, 2020). Finally, Songailaitė et al. applies modern deep learning techniques to this task. They develop their own dataset of 11,726 emails, and fine-tuned a DistilBERT, TinyBERT and RoBERTa classifier. These models represent the baseline for this classification task, with an f1 of above 0.985 (Songailaitė, 2023).

However, these models have limited usefulness to end users. The datasets these models were trained on included only general spam and ham emails (Songailaitė, 2023). As such, they would protect a user from "classic" spam emails (e.g., Nigerian princes), but not necessarily from the 100th Old Navy marketing email. Moreover, users have individual preferences for emails they would like to receive. For instance, a programmer might appreciate an email from Coursera advertising a course for the hottest new programming language, while a writer would likely consider the same email irrelevant. This work aims to move beyond the classifications of "spam" and "ham" into the classifications of "relevant" and "irrelevant." Irrelevant emails include both spam emails designed to scam their recipient and legitimate emails that do not align with the user's interests. This classification task requires us to generate a custom dataset and train unique models for each individual user. It is also more difficult than the classic "spam" vs. "ham" classification problem, as the" irrelevant" category includes both scam and legitimate emails. However, models capable of making this distinction have the potential to keep users' inboxes truly free of emails they don't want to see. In this paper, we present a proof-of-concept implementation of this system.

2 Methods

2.1 Dataset Creation

To generate individualized training data, we need emails and human-generated labels. We first generated three custom Gmail labels "Normal," "Urgent," and "Unimportant." In a real-world scenario, users would classify emails they received using these labels. For the purposes of this project, we retroactively classified the most recent 500 email chains we received. As separate emails in an email chain count as separate messages, our dataset contained 262 unimportant emails, 184 normal emails, and 90 urgent emails. We scraped the emails from these folders using the Gmail API, extracted the sender, subject, and body text (stripping attachments, images, and html code). We then remove special characters, concatenate the sender, subject, and body, assign labels, and write the results to a csv. Note the code to scrape emails and generate a dataset is general. If a user has normal, urgent, and unimportant Gmail labels, we can generate a unique dataset for them.

Note that this classification task is constrained by the amount of data available. A user may eventually have 11,000 examples across the three labels; however, they are likely to have less data initially. Email "persistence" also constrains the amount of data we have available. Email classifications too far back in time likely do not reflect a user's current preferences (e.g., the writer may have decided to switch to a programming career). Therefore, we must establish a "cutoff" email lifespan to add emails to our dataset. For this proof-of-concept system, we implement a cutoff date of the most recent two weeks via the construction of our dataset. However, a full system implementation will require a cutoff lifespan built into the scraping code.

For classification algorithms with hyperparameters we can tune, we use a 70-15-15 train/validation/test split. For algorithms with no hyperparameters (Naïve Bayes), we use an 80-20 train/test split.

2.2 Classification Algorithms

We test the classification algorithms used in previous literature, as well as several previously unexplored approaches. Note we test each classifier on the three labels (the trinary classification task), and with collapsed urgent and normal classes to represent "relevant" vs. "irrelevant" (the binary classification task).

2.2.1 Naïve Bayes

Naïve Bayes is based on Bayes Theorem, a formula that allows us to calculate the probability of an event given a condition that might be related to that event (Rish, 2001). The "naïve" assumption is that features are independent given the class label. During training, Naïve Bayes learns the probabilities of each class and the features given each class. At inference time, Naïve Bayes uses these probabilities and Bayes Theorem to calculate the probability of each class given the features for a document and selects the greatest probability class as its prediction.

We use naïve bayes with Bag-of-Words features and TF-IDF features. Bag-of-Words vectors represent the counts of all words in a document found within the training corpus (Zhang, 2010). TF-IDF values represent the importance of a word in an individual document relative to its importance across all documents in the training corpus (Ramos, 2003).

When training Naïve Bayes, we remove stop words and accents, lowercase all words, and only consider the top 7,500 features (by word frequency). For BOW features, we perform add one smoothing, and for TF-IDF features we smooth by 0.1.

2.2.2 Logistic Regression

Logistic regression represents a single layer of a neural network (McCullagh 1989). We multiply each feature by a weight and sum the results of these multiplications with a bias term (the weights and biases are learned during training via backpropagation). We then convert the resulting value to the probability a document belongs to a specific class via the softmax activation function and select the class with maximum probability. For this work, we use TF-IDF feature vectors and tune the penalty term, learning rate, and the number of training epochs. We also use early stopping (stop training once validation set performance stagnates) to avoid overfitting.

2.2.3 Support Vector Machine

A Support Vector Machine, or SVM, finds the optimal hyperplane to separate the dataset classes in their feature space (Cortes, 1995). It does so by maximizing the distance between the hyperplane and the nearest data points from each class (the margin). This generates the optimization problem of maximizing the margin while minimizing the classification error. We use TF-IDF feature vectors and tune the penalty term, learning rate, and number of epochs to train for. We again use early stopping to avoid overfitting.

2.2.4 Random Forest

Random forests are a type of ensemble model composed of decision trees (Ho, 1995). A decision tree is a set of rules about a document's features that allow the tree to make a classification decision. At each node, the tree splits the input data in the way that maximizes the class homogeneity of the resulting subsets. We repeat this process until we reach a pre-defined maximum depth, or we reach a node where all samples belong to the same class. A random forest trains multiple decision trees independently using a random sample of the data for each tree and uses the majority classification decision of the trees as its overall classification.

We tune the number of trees in the forest, the criterion to calculate homogeneity, the minimum number of samples required to split a decision tree node, and the function that determines the number of features to consider when looking for the best split of a decision tree node. We use TF-IDF feature vectors for all random forest classifiers.

2.2.5 BERT Encoder/Decoder

We can also use BERT to generate context vectors of our input (Kenton, 2019). These context vectors can be used as separate input to a decoder model that will output an email's classification. We use the BERT model's final layer output as our context vector, and our decoder consists of two linear layers and a dropout layer between the BERT encoder and the decoder. We chose the DistiliBERT base uncased model as the model to generate our context vectors. This model is more computationally performant than other BERT models (important in an actual application of this system) and retains most of the accuracy of larger BERT models (Sanh, 2019). We chose to freeze the parameters of the DistiliBERT model to avoid catastrophic forgetting (Wang, 2020). We fine-tune the number of training epochs, the learning rate, the dropout probability, and the batch size.

3 Results

We train each classification algorithm on the binary irrelevant/relevant classification task and the urgent/normal/irrelevant classification task. The binary models generally have higher f1; however, the trinary classification models potentially have more utility to the end user. In a real version of the system, allowing users to choose between binary and trinary classification is a potential feature.

3.1 Naïve Bayes

With bag-of-words features, Naïve Bayes achieves a macro averaged test f1 of 0.80 on the binary classification task and 0.69 on the trinary classification task. With tf-idf features, Naïve Bayes achieves a macro averaged test f1 of 0.92 on the binary classification task and 0.81 on the trinary classification task.

3.2 Logistic Regression

The best performing logistic regression classifier achieves a test macro-averaged f1 of 0.89 on the binary classification task with 500 training epochs, a learning rate of 0.001, and an l2 penalty term. On the trinary classification task, it achieved a test macro-averaged f1 of 0.81 with 500 training epochs, a learning rate of 0.0001, and an l2 penalty term.

3.3 Support Vector Machine

The best performing SVM classifier achieved a test macro-averaged f1 of 0.86 on the binary classification task with 500 training epochs, a learning rate of 0.0005, and an elasticnet penalty term. On the trinary classification task, it achieved a test macro-averaged f1 of 0.79 with 500 training epochs, a learning rate of 0.001, and an l2 penalty term.

3.4 Random Forest

On the binary classification task, we achieve a test macro-averaged f1 of 0.84 with 75 trees, entropy as the criterion, the feature determiner function as log base 2, and the minimum number of samples required to split a tree as 4. On the trinary classification task, we achieve a test macro-averaged f1 of 0.74 with 100 trees, gini as the criterion, the feature determiner function as square root, and the minimum number of samples required to split a tree as 3.

3.5 BERT Encoder/Decoder

We used the concatenated subject/sender/email body as input to the Encoder/Decoder model. On the binary classification task, this model achieves a test macro-averaged f1 of 0.83 with a dropout probability of 0.25, 16 training epochs, a batch size of 8, and a learning rate of 0.005. On the trinary classification task, this model achieves a test macro-averaged f1 of 0.70 with a dropout probability of 0.20, 16 training epochs, a batch size of 8, and a learning rate of 0.005.

4 Discussion

Surprisingly, Naïve Bayes with Tf-Idf features outperformed all other classification methods on the binary classification task and tied for the best performance on the trinary classification task. The BERT-based model under-performed, with the second-worst performance on both the binary and trinary classification tasks. This would indicate that context is not as informative to this classification task as the actual words within a document. As many domains of discourse are concentrated largely in a single category (e.g., as a non-dancer, most emails about dance groups and performances are irrelevant to me, while as a computer scientist, most emails about computer science research and reading groups are), the words associated with these domains of discourse also mostly belong to a single category. As a result, individual words become a good predictor of document category, and the context between them is largely irrelevant.

4.1 Error Analysis

We now present several informative examples of documents the binary Naïve Bayes with Tf-Idf features classified incorrectly.

4.1.1 Example 1

Email: *Sigma Phi Epsilon 24S PARKING AUCTION Do you want parking for the spring term? ...*
Predicted Label: irrelevant
Actual Label: relevant

This document was likely classified incorrectly due the clustering of certain domains of discourse with specific classes. As someone who is not affiliated with Greek life, I marked almost all emails from Greek houses as irrelevant. However, I might need parking in the future, so I marked this email as relevant. This email shows the Naïve Bayes approach is not good at finding these exceptions; instead, it clusters by domain of discourse.

4.1.2 Example 2

Email: *Student Government DSG Statement Clarifying Standardized Testing Requirement…*
Predicted Label: relevant
Actual Label: irrelevant

This document was misclassified for a similar reason as the previous document but with the opposite classification error. I marked most of the Dartmouth Student Government emails as relevant because I am interested in updates to Dartmouth policies and student life. However, as I am not involved in activism involving standardized testing requirements, I marked this email as irrelevant. Naïve Bayes was unable to find this exception and marked this email as relevant likely because it included "Dartmouth Student Government."

4.2 Ethical Considerations

The primary ethical concern with this work is the privacy of individual user's datasets. Were a bad actor able to gain access to a user's individual csv file, they would be able to read a user's emails and potentially gain access to confidential information. However, in an actual implementation of this system, the dataset would be stored locally on the user's computer or securely within the Google Cloud (to train a deep learning classifier via Google Collab). Still, it is important that this system not be modified in the future to scrape user data to preserve user privacy.

5 Conclusion

This work shows the viability of individualized email classifiers. Our best performing model, Naïve Bayes, achieved a test set macro-averaged f1 of 0.92 for the binary classification task, and 0.81 on the trinary classification task. Not only are these high f1 scores indicative of a useful model, but Naïve Bayes is also lightweight and performant. It represents a promising approach for a full version of this system that is useful to users without consuming excessive computational resources.

The immediate direction for future work is building a minimum viable product for this system. We currently do not have code to use a trained model to perform inference on unread emails. There are also several features we could implement that would improve user experience. Letting a user choose between the binary and trinary classification tasks or their choice of classification model would provide extra customization. Additionally, letting users set the time threshold for when emails become irrelevant could allow them to modify Inbox Guardian to suit their needs. Finally, curating a larger dataset may allow us to have more success with data-intensive models like BERT.

6 References

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*, 273-297.

Gangavarapu, T., Jaidhar, C. D., & Chanduka, B. (2020). Applicability of machine learning in spam and phishing email filtering: review and approaches. *Artificial Intelligence Review*, *53*, 5019-5081.

Ho, T. K. (1995, August). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition* (Vol. 1, pp. 278-282). IEEE.

Kenton, J. D. M. W. C., & Toutanova, L. K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT* (Vol. 1, p. 2).

McCullagh, P., & Nelder, J.A. *Generalized Linear Models*. New York: Chapman and Hall.

Mohammed, S., Mohammed, O., Fiaidhi, J., Fong, S., & Kim, T. H. (2013). Classifying unsolicited bulk email (UBE) using python machine learning techniques. *International Journal of Hybrid Information Technology*, *6*(1), 43-56.

Ramos, J. (2003, December). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, No. 1, pp. 29-48).

Rish, I. (2001, August). An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence* (Vol. 3, No. 22, pp. 41-46).

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Songailaitė, M., Kankevičiūtė, E., Zhyhun, B., & Mandravickaitė, J. BERT-Based Models for Phishing Detection.

Wang, Y., Yao, Q., Kwok, J. T., & Ni, L. M. (2020). Generalizing from a few examples: A survey on few-shot learning. ACM computing surveys (csur), 53(3), 1-34.

Zhang, Y., Jin, R., & Zhou, Z. H. (2010). Understanding bag-of-words model: a statistical framework. *International journal of machine learning and cybernetics*, *1*, 43-52.