Artificial intelligence-based spam filtering using a neuro-linguistic approach with PyTorch framework

Balázs Tóth
John von Neumann Faculty of Informatics
Óbuda University
Budapest, Hungary
mwzx0d@stud.uni-obuda.hu

Valéria Póser
Biomatics and Applied Intelligence Institute
Óbuda University
Budapest, Hungary
poser.valeria@nik.uni-obuda.hu

Szandra Anna Laczi
Doctoral School of Applied Informatics and Applied Mathematics
Óbuda University
Budapest, Hungary
laczi.szandra@nik.uni-obuda.hu

***Abstract*—This paper outlines the creation of an artificial intelligence spam filter using PyTorch, grounded in neurolinguistic approaches, specifically natural language processing (NLP). The developed model efficiently identified and filtered messages that exhibited suspicious characteristics.**

***Keywords*—PyTorch; development; natural language processing; spam filtering; security**

I. Introduction

Digital communication has become almost indispensable in people's daily lives. Unfortunately, spam is growing exponentially at the same time, challenging the systems that filter out unwanted content. It is critical that the software we use to send and receive message filters them reliably.

This paper shows how the PyTorch framework and natural language processing approaches can be used together to design an intelligent spam filtering system. It is able to recognise if the given data is general or suspicious message.

A. Typical patterns in email spam

- Phishing emails are designed to impersonate a trusted organisation and lure recipients into revealing sensitive information such as usernames, passwords or any valuable data.
- Malware emails send attachments or links to malicious software designed to infect the recipient's device with viruses, ransomware or other malicious programs.
- The advance payment scam, these emails promise large sums of money in exchange for a small advance.
- Fake lottery or prize scams, which falsely claim that recipients have won a lottery prize and often ask for personal details or payment to claim the alleged prize.
- Survey emails designed to collect personal data for fraudulent purposes.

These are the most common types of email spam but the list could be endless.

II. Why PyTorch?

In 2016, Facebook's AI research group, now known as Meta, took the lead in creating the PyTorch framework and generously shared it with the global community as an open-source resource. PyTorch has gained recognition for its outstanding qualities, being praised for its exceptional simplicity, impressive flexibility, and inherent efficiency. These remarkable features have solidified PyTorch's position as a fundamental and highly regarded tool in the fields of artificial intelligence and machine learning [1].

TABLE I: Comparing PyTorch with Keras

Category	PyTorch	Keras
API Level	Low	High
Datasets	Large datasets, high-performance	Smaller datasets
Debugging	Good debugging capabilities	Challenging
Pretrained models	Yes	Yes
Speed	Fast, high-performance	Slow, low-performance
Written in	Lua	Python
Visualization	Limited	Depends on backend

Table I provides a detailed comparison between the PyTorch and Keras frameworks [2]. The key factor influencing the choice of PyTorch is its impressive performance and the ability to handle large datasets seamlessly. This pivotal decision is grounded in the framework's robust capabilities, making it a reliable choice for our study.

III. Overview of spam statistics

As discussed previously about what are the common patterns in email spams, it is crucial to know about the statistics too. These statistical values show that the companies and end users are not having the of knowledge how to handle potential harmful messages in their inbox if that is not handled by the spam filter. End users may not know either that their personal data is valuable for the hackers, such as their name, birthday or address might be enough to steal identities.

TABLE II: Spam statistics [3]

Threat actors	In 2023, 32% of threat actors used email as a pathway to disrupt organisations.
Grow	The total number of business and consumer emails sent and received daily will exceed US$333 billion in 2022 and is forecast to grow to over US$392 billion by year-end 2026.
Global value	In 2022, nearly 49% of all e-mails globally were identified as spam, up from 46% in 2021.
Origin	The United States of America currently leads as the country of origin of spam emails with 8,765 spam emails sent.
Report	In 2023, 18 million emails were reported by the State of the Phish organisations over 12 months.
Most common form	Phishing is the most common form of cybercrime, with an estimated 3.4 billion spam emails sent every day.
Accounts	Spam accounts for 14.5 million messages globally per day. This makes up 45% of all emails according to research.
Compromise	Scams and fraud comprise only 2.5% of all spam emails, however, identity theft makes up 73% of this figure.
Malware threats	In 2022, the number of unknown malware threats spiked to 3.8 million, indicating a substantial 46% surge according to Trend.
Spam blocks	According to Google, Gmail blocks more than 100 million spam emails per day.

IV. Methodology

A. Dataset description

In this case, the dataset consists of a collection of text messages paired with the corresponding class labels after preprocessing, which can diffuse from each dataset. Table III shows the structure of the dataset, serving as an illustrative example. It is important to note that the training is not based on this example data, see Table III. Various datasets with diverse data will be employed subsequently, making Table III a representative instance rather than an exhaustive depiction of the training process.

TABLE III: Dataset

Text	Class
Hey, how's it going?	Not spam
Pay now, or your account will be locked!	Spam
Congratulations! You have won a free vacation!	Spam
Have a great day!	Not spam
...	...

```
class SpamClassifier(nn.Module):
  def __init__(self, input_dim,
      hidden_dim1, hidden_dim2, output_dim
      , dropout_prob):
      super(SpamClassifier, self).__init__
          ()
      self.fc1 = nn.Linear(input_dim,
          hidden_dim1)
      self.fc2 = nn.Linear(hidden_dim1,
          hidden_dim2)
      self.fc3 = nn.Linear(hidden_dim2,
          output_dim)
      self.dropout = nn.Dropout(p=
          dropout_prob)

  def forward(self, x):
      x = F.relu(self.fc1(x))
      x = self.dropout(x)
      x = F.relu(self.fc2(x))
      x = self.dropout(x)
      x = self.fc3(x)
      return x
```

Fig. 1: PyTorch code of the model architecture

B. Model architecture

The model's code in figure 1 followed a feedforward architecture with three fully connected layers. These layers, denoted as $fc1$, $fc2$, and $fc3$, were responsible for the linear transformation [3] of the input data. The first layer processes the input features and the subsequent layers extract hierarchical representations, leading to the final output layer.

Rectified Linear Units ($ReLU$) [4] serve as activation functions applied after the linear transformations in the first and second layers. To prevent overfitting, a dropout [5] layer was strategically placed after each Rectified Linear Unit ($ReLU$) activation step. This layer randomly sets a fraction of the output units to zero during training, ensuring that the network generalizes well to the unseen data.

The output of the model, which represents the likelihood of spam or non-spam for binary classification, is produced by the final layer $fc3$, without any activation function applied to it.

C. Used parameters

For hidden dimension one, set to 64, this parameter determines the number of neurons in the first hidden layer of the neural network. For hidden dimension two, we assigned a value of 32, which defines the number of neurons in the second hidden layer. This layer builds on the representations learned by the first layer and extracts additional features from the data. Output dimension configured as 2, which defines the number of classes in the classification task. With a dropout probability of 0.5, dropout layers are incorporated after each activation function to mitigate overfitting. To achieve improved accuracy, the optimal number of epochs may vary for each dataset. The choice of epochs balances the training in time with the convergence of the model to a stable state. The test size, set to 0.2, determines the proportion of the dataset reserved for testing the model's performance. A test size of 0.2 implies that 20% of the data is held out for evaluation, while the remaining 80% is used for training. The learning rate specified as 0.001, governs the step size in the gradient descent optimization process.

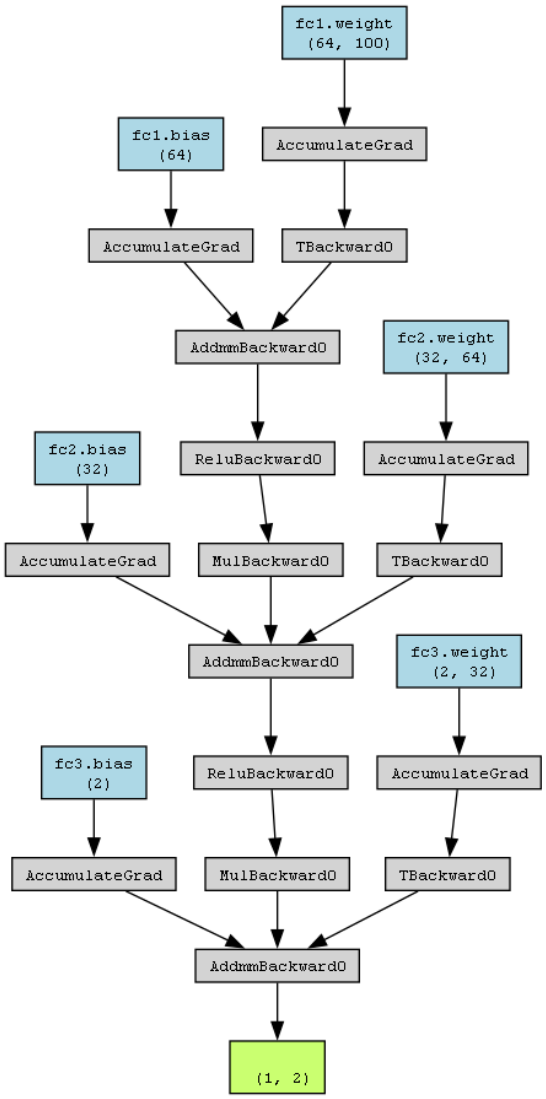

Fig. 2: Visualization of the model

D. Training process

Figure 2 shows an example input dimension (100) to present the entire training process using the model discussed previously. It is noteworthy that the input dimensions can differ for each dataset.

The Cross Entropy Loss ($nn.CrossEntropyLoss()$ as shown in figure 1 was employed [8]. This loss function is well-suited for multi-class classification tasks, providing a measure of the disparity between predicted class probabilities and the true distribution of classes.

The Adam optimizer ($optim.Adam$ as shown in figure 1 is selected for its adaptive learning rate properties [9].
Adam is expected to navigate the model parameters effectively during the training process.

True Positives (TP):

- Cases that were positive (spam) and correctly predicted by the filter were positive (spam).
- Example: An email containing known spam keywords and characteristics is correctly classified as spam by the filter.

True Negatives (TN):

- Cases that were indeed negative (not spam), and the filter correctly flagged them as negative (not spam).
- Example: A regular non-spam email, without any spam-like attributes, is correctly identified as not spam by the filter.

False Positives (FP):

- Cases that were negative (not spam) but wrongly flagged by the filter as positive (spam).
- Example: A legitimate email from a friend contains certain words or patterns that the spam filter misclassifies as spam.

False Negatives (FN):

- Cases that were actually positive (spam) but the filter incorrectly flagged them as negative (not spam).
- Example: An actual spam email manages to evade detection and is incorrectly classified as a non-spam email.

E. Evaluation metrics

These mathematical formulas shows how the accuracy, precision, recall (sensitivity) and f1 score are calculated [6].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy measures the overall correctness of the model by calculating the proportion of correctly predicted cases (true positives and true negatives) relative to all cases.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision is the ratio of correctly predicted positive observations (true positives) to the total number of predicted positives. It focuses on how many of the predicted positive cases were actually positive.

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN}$$

The recall calculates the ratio of correctly predicted positive observations (true positives) to the total number of true positives. This shows how well the model captures all positive cases.

$$\text{F1 score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The F1 score is the harmonic mean of accuracy and recall. It balances accuracy and recall and provides a single metric for evaluating model performance. It is particularly useful when the distribution of classes is uneven.

V. Results

1. Spam assassin dataset: Provided by Apacha Spam Assassin Public Corpus, where the dataset contains 5796 unique values [10]. Has 1896 spam messages and 3900 not spam messages. The results are shown in Table IV, including precision, recall, f1 core and average values.

TABLE IV: Spam Assassin dataset

	Precision	Recall	F1 score
Not spam	0.99	1.0	0.99
Spam	1.00	0.97	0.99
Macro average	0.99	0.99	0.99
Weighted average	0.99	0.99	0.99

2. SMS Spam dataset: Provided by UCI Machine Learning Repository where the dataset contains 5573 unique values. [11] Has 4825 spam messages and 748 not spam messages. The results are shown in Table V, including precision, recall, f1 score and average values.

TABLE V: SMS Spam dataset

	Precision	Recall	F1 score
Not spam	0.98	1.0	0.99
Spam	0.98	0.88	0.92
Macro average	0.98	0.94	0.96
Weighted average	0.98	0.98	0.98

3. YouTube spam dataset: Provided by UCI Machine Learning Repository where five datasets can be found, our project merged them. This means that the merged dataset contains 1961 unique values [12]. Has 1012 spam messages and 949 not spam messages. The results are shown in Table VI, including precision, recall, f1 score, and average values.

TABLE VI: YouTube spam dataset

	Precision	Recall	F1 score
Not spam	1.0	0.96	0.98
Spam	0.98	1.00	0.99
Macro average	0.99	0.98	0.98
Weighted average	0.98	0.98	0.98

Table VII presents the overall accuracy of the three public datasets for a better overview of which the model was trained.

TABLE VII: Overall accuracy values by each dataset

	Accuracy
Spam assassin dataset [10]	99.14
SMS spam dataset [11]	98.12
YouTube spam dataset [12]	98.57

VI. Conclusion

In our project, we developed an artificial intelligence-driven spam filter solution with PyTorch framework that has demonstrated its effectiveness through training on various publicly available datasets. Possible further developments and improvements:

In addition to text-based filtering, image filtering plays a crucial role in categorizing visual content and enhancing the overall security of message processing.

The integration of adaptive learning techniques and optimizations, such as employing learning rate schedulers, enhances the attainment of optimal results by promoting the continuous refinement of the filtering models. Adaptive learning allows the system to dynamically adjust to evolving patterns in spam content, whereas optimizations such as learning rate schedulers fine-tune the training process, leading to improved model convergence and increased accuracy in spam classification.

References

[1] *PyTorch, https://pytorch.org/*, Last viewed; 17:47, 31th of December, 2023

[2] *PyTorch vs Tensorflow vs Keras*, https://www.datacamp.com/tutorial/pytorch-vs-tensorflow-vs-keras, Last viewed; 20:59, 6th of December, 2023

[3] *Linear — PyTorch 2.1 documentation.*, https://pytorch.org/docs/stable/generated/torch.nn.Linear.html, Last viewed; 18:23, 31th of December, 2023

[4] *ReLU — PyTorch 2.1 documentation.*, https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html, Last viewed; 18:26, 31th of December, 2023

[5] *Dropout — PyTorch 2.1 documentation.*, https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html, Last viewed; 18:33, 31th of December, 2023

[6] D. Sharma and A. Sharaff, *Identifying Spam Patterns in SMS using Genetic Programming Approach*, 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 396-400

[7] *Spam statistics: a deep dive into unwanted emails*, https://eftsure.com/statistics/spam-statistics/, Last viewed; 18:34, 6th of December, 2023

[8] *CrossEntropyLoss — PyTorch 2.1 documentation.*, https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, Last viewed; 09:03, 13th of December, 2023

[9] *Adam — PyTorch 2.1 documentation.*, https://pytorch.org/docs/stable/generated/torch.optim.Adam.html, Last viewed; 09:07, 13th of December, 2023

[10] *Email classification. (2021, August 6).*, https://www.kaggle.com/datasets/ganiyuolalekan/spam-assassin-email-classification-dataset, Last viewed; 09:53, 13th of December, 2023

[11] *SMS spam collection - UCI Machine Learning Repository.*, https://archive.ics.uci.edu/dataset/228/sms+spam+collection, Last viewed; 10:45, 13th of December, 2023

[12] *YouTube spam collection - UCI Machine Learning Repository.*, https://archive.ics.uci.edu/dataset/380/youtube+spam+collection, Last viewed; 08:07, 14th of December, 2023