Spam Filtering

Farhan Shoukat, Kamal Subhani, Hassan Alvi

Email: [l154292@lhr.nu.edu.pk](mailto:l154292@lhr.nu.edu.pk), [l154048@lhr.nu.edu.pk](mailto:l154048@lhr.nu.edu.pk), [l154203@lhr.nu.edu.pk](mailto:l154203@lhr.nu.edu.pk)

***Abstract*—In many applications like Gmail, Facebook, WhatsApp etc. spam filtering is very important. This project presents kNN, Artificial Neural Network and Naive Bayes to classify email as spam or not spam. Our experimental results demonstrate the 97% accuracy with Naive Bayes and 97.24% with Artificial Neural Network. kNN did not gave results in time.**

***Keywords—email spam estimation; kNN, Neural Network, Naive Bayes; email spam classification, email***

# Introduction

Many real world applications require spam filtering system; for example, Gmail, Facebook, Twitter, WhatsApp require a spam filtering system to protect their users. In this project, we found that spam filtering can be achieved using Naive Bayes and Artificial Neural Network. Both performed quite well with respect to time and accuracy. Naive Bayes gave very good results in no time. Neural Network provided better results with a little bit more time.

This article is framed as follows. Section II reviews Related Work, Section III describes the image processing technique and neural network architecture, Section IV explain the experimental setup, dataset and results. Section V conclude this document.

# Literature Review

1. **An Evaluation of Naive Bayesian Anti-Spam Filtering [1]**

This paper evaluated Naive Bayes algorithm for email spam filtering. Each email is converted to a vector where each value in a vector represents whether a word is present in the email or not. The probability of each word is calculated from training data. To predict whether an email is spam or not spam, the probability of both classes {spam, not spam} is calculated. Then depending on the precedence/weight the class is selected.

1. **Boosting trees for anti-spam email Filtering[2]**

This paper describes a set of comparative experiments for the problem of automatically filtering unwanted electronic mail messages. Several variants of the AdaBoost algorithm with confidence– rated predictions have been applied, which differ in the complexity of the base learners considered.

1. **A review of machine learning approaches to Spam filtering[3]**

* Tokenization, which extracts the words in the message body
* Lemmatization, reducing words to their root forms (e.g., ‘‘extracting” to ‘‘extract”)
* Stop-word removal, eliminating some words that often occur in many messages (e.g., ‘‘to”, ‘‘a”, ‘‘for”)
* Representation, which converts the set of words present in the message to a specific format required by the machine learning algorithm used**.**

The procedure that was discussed in [1] was also discussed in this paper

.

# Methodology

In this project, We will use Naive Bayes Classifier, kNN and Artificial Neural Network to classify an email as spam or not spam.

**1- Email Processing**

The email is first converted into a string. Then all characters are converted to lowercase and all numbers and characters that do not have any meaning are removed. Then email is converted to a scipy[4] sparse vector using TfidfVectorizer or CountVectorizer from scikit-learn[5] library. While converting into sparse vector stop words are removed and email is stemmed using porter stemmer from nltk[6] library. Email can also be normalized or scaled using scikit-learn[5].

Input Email

Convert to string

Convert to lowercase

Remove numbers and special characters

Remove stop words

Convert to sparse vector

Figure 2. Email processing steps

**2- Classification**

Naive Bayes classifier, kNN and Artificial Neural Network takes an email body as input and assign it a class from the list of classes shown in Table 1

|  |  |
| --- | --- |
| **Table 1. List of classes** | |
| **Id** | **Label** |
| 0 | Ham/Legitimate/not spam |
| 1 | Spam |

**3a- Naive Bayes Architecture**

In order to design Naive Bayes classifier, this project used scikit-learn[5], a free software machine learning library for the Python. Using scikit-learn, Naive Bayes is designed using either of MultinomialNB or BernoulliNB present in scikit-learn.

**3b- Neural Network Architecture**

We have utilized MLPClassifier from scikit-learn[5]. This library stores multiple hidden layers which uses back propagation to solve the problem. This is multilayered perceptron model. We have experimented with different solvers like ‘lbfgs’,’adam and ‘sgd’. The best results were obtained from the solver ‘lbfgs’. Whilst the activation function we have used to manipulate the neural network is ‘relu’.

We also tried tensorflow[6] Neural Network but the due to large data size it was giving memory overflow exception. In this network we used 4 hidden layers with different units and “relu” activation function.

**3c- KNN Architecture**

We gave more than 3 hours to run the program with kNN but it did not gave an output.

# Experimental Setup And Results

# Dataset

In this project, subset of Enron Email Dataset provided by Enron Cooperation was used. This subset consists of 33687 emails out of which 16545 are not spam/ham and 17142 are spam. We divided this dataset for training and testing as follows

|  |  |  |
| --- | --- | --- |
| **Table 3: Number of Emails in training data set and testing data set** | | |
| **Email Type** | **Training Dataset** | **Testing Dataset** |
| Not spam | 11545 | 5000 |
| spam | 12142 | 5000 |

# Results

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 4: Results for Naive Bayes** | | | |
| **Parameters** | **Accuracy** | **Wrongly Identified Legitimate as Spam** | **Wrongly Identified Spam as Legitimate** |
| TfidfVectorizer, MultinomialNB | 96.81 | 1.14 | 2.05 |
| CountVectorizer, MultinomialNB | 95.8 | 2.07 | 2.13 |
| TfidfVectorizer, BernoulliNB | 91.54 | 0.28 | 8.18 |
| CountVectorizer, BernoulliNB | 91.54 | 0.28 | 8.18 |
|  |  |  |  |
| TfidfVectorizer, MultinomialNB, Scaled | 91.89 | 1.18 | 6.93 |
| CountVectorizer, MultinomialNB, Scaled | 91.36 | 1.51 | 7.13 |
| TfidfVectorizer, MultinomialNB, normalized | 96.81 | 1.14 | 2.05 |
| CountVectorizer, MultinomialNB, normalized | 96.62 | 1.31 | 2.07 |
| Both scaling and normalization had no effect in case of BernoulliNB for both vectorizers | | | |
|  |  |  |  |
| TfidfVectorizer, MultinomialNB, alpha=0.2 | 97 | 1.09 | 1.91 |
| TfidfVectorizer, MultinomialNB, alpha=0.3 | 96.98 | 1.11 | 1.91 |
| TfidfVectorizer, MultinomialNB, alpha=0.4 | 96.97 | 1.09 | 1.94 |
| TfidfVectorizer, MultinomialNB, alpha=0.6 | 96.95 | 1.1 | 1.95 |
| TfidfVectorizer, MultinomialNB, alpha=0.8 | 96.85 | 1.12 | 2.03 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 5: Results for Artificial Neural Network** | | | |
| **Parameters** | **Accuracy** | **Wrongly Identified Legitimate as Spam** | **Wrongly Identified Spam as Legitimate** |
| TfidfVectorizer, lbfgs, alpha=1e-5,(15,15) | 96.1 | 1.25 | 2.65 |
| TfidfVectorizer, lbfgs, alpha=100,(15,15) | 96.76 | 2.18 | 1.06 |
| TfidfVectorizer,sgd, alpha=100,(15,15) | 50.0 | 50.0 | 0 |
| TfidfVectorizer,lbfgs, alpha=0.1,(5) | 97.24 | 1.03 | 2.01 |
| TfidfVectorizer,lbfgs, alpha=0.1,(50,50,50) | 96.8 | 1.73 | 2.31 |
| CountVectorizer, lbfgs, alpha=1000(15,15) | 95.11 | 3.9 | 0.99 |

# Conclusion

In this project we have used Naive Bayes, KNN and Artificial Neural Networks. Naive Bayes gave very good results with accuracy of 97% and it is very fast too. In artificial neural networks , firstly MLP Classifier from scikit-learn provided the best accuracy i.e. 97.24% while the tensorflow neural network didn’t work due to large data size and memory requirement. Moreover, we also tried KNN but it kept running for more than 3 hours so it is not feasible for real world perspective.

##### **References**

1. I. Androutsopoulos, J. Koutsias, Konstantinos V. Chandrinos, G. Paliouras and Constantine D. Spyropoulos, *An Evaluation of Naive Bayesian Anti-Spam Filtering*, arXiv, 2000.
2. Carreras, X., & Marquez, L, *Boosting trees for anti-spam email filtering*, arXiv preprint CS/0109015, 2001.
3. Thiago S. Guzella, Walmir M. Caminhas, *A review of machine learning approaches to Spam filtering*, Elsevier, 2009.
4. Community library project, *Scipy,* Available at: <https://www.scipy.org/> (Accessed: 1st May 2018).
5. David Cournapeau, *Scikit-learn,* Available at: <http://scikit-learn.org/stable/> (Accessed: 1st May 2018).
6. Qazi Qayyum Nawaz, *Nltk*, Available at: <https://www.nltk.org/> (Accessed: 1st May 2018).
7. Google Brain Team, *TensorFlow*, Available at: <https://www.tensorflow.org/> (Accessed: 5th May 2018).