Spam Classification

Yang Xuze Zhao Qiuran Feng Yifan

Yang Jialang Peng Yuxiang

South China University of Technology

November 6, 2020

Abstract

With the development of science and technology, the increasing popularity of the Internet, and its convenience and low cost, e-mail has been used by more and more criminals to maliciously spread commercial advertisements, computer viruses and bad information. Many email services today provide spam filters that are able to classify emails into spam and non-spam email with high accuracy. From which we can see the importance of a high efficiency spam classifier. Proceeding from that, our team started the spam classification project and successfully created high accuracy classifiers to distinguish spam from the received e-mails.

I. Introduction

E-mail is a special text containing text, images and videos. It has become a tool for communication on the Internet, whereas there are quantities of disturbing spams in our Inbox. A large number of statistics and research reports show that spam accounts for more than 50% of the world's mail, to people's life, work has brought interference, and waste a lot of network bandwidth. To improve the accuracy of spam filtering and ensure the security of information has aroused people's wide concern.

The low operating cost of the mail system and the expansion of its user groups have attracted a large number of users. The simple mass mailing technology and the E-mail address collected on the Internet can be used to realize the mass mailing of spam, which results in the mail system has to consume additional system resources to deal with a large number of junk mails. Imagine that if we need to spend some time to distinguish e-mails every day, our work efficiency and mood will be affected. Therefore, automatically identifying spam is of great significance and application value. In order to solve this problem, we need to develop a solution on software, design a low-cost mail filtering

system into mail screening, reduce the operation cost of the system.

Our project is aiming to create a classifier to identify the received e-mails based on many classification methods. Besides, we will try our best to improve the accuracy, precision and recall to be higher than 90%.

II. Formalization

Procedures

Firstly, we had a discussion about what project should we establish and we finally chose this spam classification project. Next, we debated on the methods, we searched the related information and then decided to use naïve bayes and other two classifiers to work as the base classifier. The next four days we mined the dataset from the Internet and found quantities of useful dataset. Then comes the important part, we cooperated to finish the programming step, implementing what we had learnt and looking up for relevant information, with multiple times of debugging, we finally reached the goal that accuracy, precision and recall to be higher than 90%. Afterwards, we evaluated the results and assured that the results are pleasant.

Work Distribution

Task	**People**
Collecting and processing data	Zhao Qiuran, Yang Jialang, Peng Yuxiang
Writing program	Zhao Qiuran
Evaluating program	Feng Yifan
Writing report	Yang Xuze, Zhao Qiuran.

Programming Requirements

- Anaconda 3 with Python 3.7
- Python libraries including sklearn, pandas, time, matplotlib, joblib, nltk, re, seaborn
- PyCharm used as IDE

Datasets

https://www.kaggle.com/uciml/sms-spam-collection-dataset

https://www.kaggle.com/shravan3273/sms-spam

They totally contain about 11000 samples, and both are .csv files, so we put it into one file. Figure II-1 shows the ham vs. spam proportion

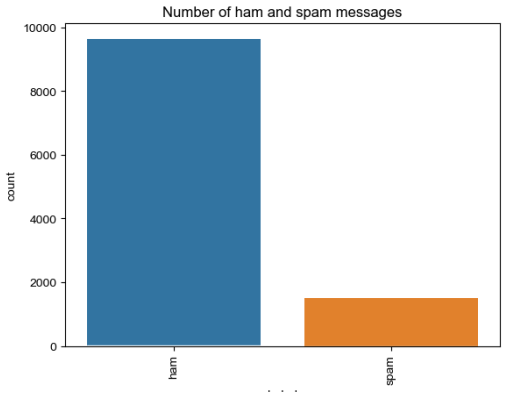

Figure II-1

III. Algorithms

Overall description

Several classification algorithms are used in the program and realized through python. We first went through all the data, use the re library, define a function that can process the content of the emails and make the detecting dictionary from processed ones. We get a vocabulary list which contains 6709 words. Then we will use the processed emails and the vocabulary list to construct feature vectors for training. Having done all the feature vectors, we used Naïve bayes (Multinomial), Support Vector Machine, Multi-layer Perceptron, Decision Tree and k-Nearest Neighbor to train the processed data set. All classification algorithms are realized through sklearn library in python.

Processing Data

There are many things we don't need and may disturb the results in the emails such as HTML tags, email address, URL and so on, which will disturb the classification of the emails, so we will first use the re library which can delete or substitute distractions through regular expressions, for example, we can delete HTML tags like this:

re.sub ('< [^<>_] +>', '', content)

then the HTML tags will be replaced by a blank space, we do the same thing on numbers, email address and URL, they will be replaced by 'number', 'emailaddr', 'httpaddr' After a simple processing, the complex email content will be transformed into text containing only pure words, then another powerful tool will be used to deal with more details. We will import nltk, which is a powerful Natural Language Processing library in python, we can use the nltk. stem. porter to transform words into its archetype, for example, 'driving' and 'drove' will be transformed into 'drive', which can effectively reduce the number of words with duplicate meanings in the word list and improve the efficiency of training. After all this was done, the content will be transformed into single word vectors, and we use the for loop to traverse them and compare each one with the words in the vocabulary list and create their own feature vectors like the following:

$$x = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^n.$$

Where x [0] =0 for spam, 1 for ham, n= (length of vocabulary+1), if the word in an email exist in the vocabulary, then the corresponding position in the vector will be 1. Then we put all the vectors into a list and save it as a train_set.csv file for training.

Naïve Bayes (Bernoulli NB)

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. It uses Bayes' theorem to first find the joint probability distribution and then the conditional probability distribution. Simplicity here refers to the assumption of conditional independence in the calculation of likelihood estimation. The basic principle can be given by the following formula:

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)}$$

Where P(Y|X) is the posteriori probability, P(Y) is the prior probability, P(X|Y) is the likelihood probability, P(X) is the evidence.

In this project, because of the sparse multivariate discrete values and the binary characteristics of the features, we choose Bernoulli NB for training. In Bernoulli model, for a feature vector like the following:

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \text{a} \\ \text{aardvark} \\ \text{aardwolf} \\ \vdots \\ \text{buy} \\ \vdots \\ \text{zygmurgy} \end{matrix}$$

We need to build a model for p(x|y), Suppose that given classification y, each component of the eigenvector x_i is conditionally independent. In other words, the naive Bayes hypothesis in the context of text classification is that the occurrence of one word in the text does not affect the probability of the occurrence of other words in the text. So we have

$$p(x_1, \dots, x_n|y)$$
$$= p(x_1|y)p(x_2|y, x_1)p(x_3|y, x_1, x_2) \dots p(x_n|y, x_1, \dots, x_{n-1})$$
$$= p(x_1|y)p(x_2|y)p(x_2|y) \dots p(x_n|y)$$
$$= \prod_{i=1}^{n} p(x_i|y)$$

The model function involves

$$\emptyset_{i|y=1} = p(x_{i=1}|y = 1)$$
$$\emptyset_{i|y=1} = p(x_{i=1}|y = 0)$$
$$\emptyset_y = p(y = 1)$$

The logarithm function of the maximum likelihood function is

$$\mathcal{L}(\phi_y, \phi_{i|y=0}, \phi_{i|y=1}) = \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}).$$

The maximum likelihood estimate of the parameter is obtained:

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}$$
$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}}$$
$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}$$

Then for a new sample, we can use the formula to calculate its probability:

$$p(y=1|x) = \frac{p(x|y=1)p(y=1)}{p(x)}$$
$$= \frac{(\prod_{i=1}^{n} p(xi|y=1))p(y=1)}{(\prod_{i=1}^{n} p(xi|y=1))p(y=1) + (\prod_{i=1}^{n} p(xi|y$$

But there exists a problem, assume that a word in an email that has never appeared in an email before, at 35,000 in the dictionary, gives a maximum likelihood estimate of

$$\phi_{35000|y=1} = \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)} = 1 \wedge y^{(i)} = 1\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}} = 0$$
$$\phi_{35000|y=0} = \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)} = 1 \wedge y^{(i)} = 0\}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}} = 0$$

Which means that if a word has appeared in both previous spam and non-spam e-mails, then the naive Bayesian model assumes that the word has a zero probability of appearing in any E-mail. If, the message is spam, but the formula is:

$$p(y=1|x) = \frac{\prod_{i=1}^{n} p(x_i|y=1)p(y=1)}{\prod_{i=1}^{n} p(x_i|y=1)p(y=1) + \prod_{i=1}^{n} p(x_i|y=0)p(y=0)}$$
$$= \frac{0}{0}.$$

That's not a very reasonable result, because we can't say that the probability of an event happening is 0 just because it hasn't happened in the past. So Laplace Smoothing is a commonly used smoothing method. Smoothing methods exist to solve zero probability problems. We plus λ in the numerator (λ is the Laplace smoothing, whose value is often defined as 1.), plus the number of variables in the denominator. Therefore, in the Naive Bayes problem, after corrected by Laplace Smoothing,

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 1\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} + 2}$$
$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1 \wedge y^{(i)} = 0\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} + 2}$$

We will then put it into the formula we build before and calculate all the probability (Y=0, 1), then compare which one is larger, then which one is the classification result.

Support Vector Machine (SVM)

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used for classification problems like the one in this project. An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (Figure III-1).

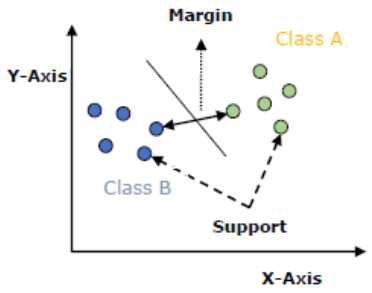

Figure III-1

Maximum margin classifier only depends on few samples, called Support Vectors. SVM

uses a technique called kernel in which kernel transforms a low dimensional input space into a higher dimensional space, which means that it converts non-separable problems into separable problems by adding more dimensions to it. In this project, we choose linear kernel:

$$K(x, x_i) = sum(x * x_i)$$

Multi-layer Perceptron (MLP)

A multilayer perceptron is a feedforward Artificial Neural Network (ANN) that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input and output layers. It can solve more complex and nonlinear problems. Figure III-2 represents three layers of perceptron with three inputs, two outputs and a hidden layer of four neurons.

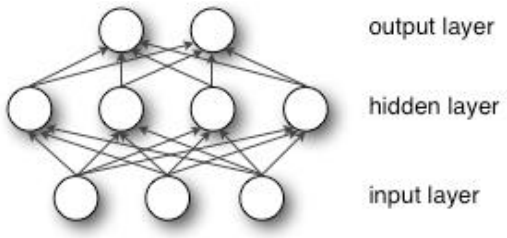

Figure III-2

All neurons in MLP are similar. Each of them has several input links (it takes the output values of several neurons in the previous layer as input) and several output links (it passes the response to several neurons in the next layer). The values retrieved from the previous layer are added to some weights, and a bias term is added to each individual neuron. Using activation functions to transform the sum, activation function can also be different for different neurons. The neuron of the hidden layer is fully connected to the input layer. The whole MLP model is summed up in a formula:

$$f(x) = G(b_2 + W_2(s(b_1 + W_1)))$$

with bias vectors b_1, b_2; weight matrices W_1, W_2 and activation functions G and S. Assuming that the input layer is represented by vector x, $h(x)=s(W_1x+ b_1)$ constitutes the hidden layer, the function s can be the sigmoid function or tanh function, but for the data in this project, a simple linear function $f(x) = x$ can do as well as these complex functions and cost less time. The output vector is then obtained as: $o(x)=G(b_2+W_2h(x))$, G represents the Softmax function. So, all the parameters of an MLP are the connection weights and biases between the layer. For a specific problem, solving the best parameter is an optimization problem. Stochastic Gradient Descent (sgd) and Adam Optimizer are two ways to solve this, here we choose Adam for its higher efficiency and ability to process large-scale data.

Decision Tree

Decision tree is the most powerful and

popular tool for classification and prediction. A decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. Three main steps are included in creating a decision tree: feature selection, generation and pruning. For node splitting, the feature with the highest information gain can be selected as the test feature, entropy and gini index can be used to calculate information gain, in this project we choose entropy. Assume that there are m classes in set D, |D| and $|C_{i,D}|$ are the number of samples in them, the information entropy of D is given by

$$\boldsymbol{Info(D) = -\sum_{i=1}^{m} p_i log_2 p_i}$$

the information gain is given by

$$\boldsymbol{Gain(C) = Info(D) - Info_C(D)}$$

The feature with the largest information gain will be chosen as the splitting feature. The basic algorithm is as following:

1. If all tuples in D belongs to the same class C_j Add a leaf node labeled as Cj

Return // Termination condition

2. Select an attribute A_i

3. Partition D = {D_1, D_2, …, D_p} based on p different values of A_i in D

4. For each $D_k \epsilon D$

Create a node and add an edge between D and D_k with label as the Ai 's attribute value in D_k

5. For each $D_k \epsilon D$

BuildDT(D_k) // Recursive call

6. Stop

In this project, a machine learning library in python called sklearn is used to do the algorithm, the parameters are all default except for the feature selection criteria which use entropy.

k-Nearest Neighbor (k-nn)

A k-nearest-neighbor algorithm is an approach to data classification that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in. An example is shown in Figure III-3

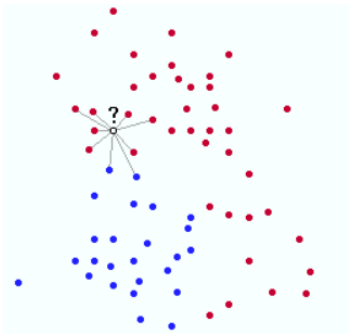

Figure III-3

It is an example of a "lazy learner" algorithm, meaning that it does not build a model using the training set until a query of the data set is performed. The general procedure for k-nn is:

Let m be the number of training data samples.

Let p be an unknown point.

1. Store the training samples in an array of data points arr[]. This means each element of this array represents a tuple (x, y).

2. for i=0 to m:

Calculate the distance d(arr[i], p).

3. Make set S of K smallest distances obtained. Each of these distances corresponds to an already classified data point.

4. Return the majority label among S.

In k-nn algorithm, by calculating the distance between objects as the non-similarity index between each object, the matching problem between objects is avoided. Euclidian distance or and Manhattan distance are generally used, here we choose Euclidian distance for the dimension of data is not that high:

$$d(x, y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

The value of k is important, it can be determined through cross-validation (the sample data is divided into training data and validation data in a certain proportion). Starting from a small value of k, and increased continuously, then the variance of the validation set is calculated to finally find an appropriate value of k, python can be used to realize it, the plot can be drawn like Figure III-4

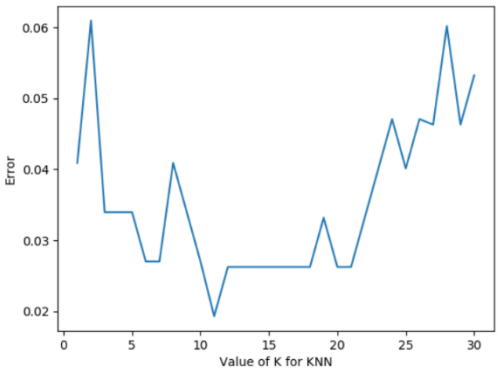

Figure III-4

Obviously, the error is the smallest when k is 11, so take k=11. In sklearn library, we use KNeighborsClassifier with parameters k=11, weights='distance', algorithm='kd_tree', others by default.

Test.py

In this program, the classifiers trained before will be loaded to do predictions of the emails received (also get processed by the same method as the train data), we make a simple integration algorithm for the 5 classifiers. As we all know, there will be 5 results produced by the classifiers respectively, if they produce different results, we will choose the one which is produced by more classifiers.

IV. Results

For spam

	Precision	Recall	F1-measure
BNB	0.99	0.94	0.97
SVM	1.00	0.98	0.99

MLP	1.00	0.98	0.99
DT	0.97	0.98	0.98
KNN	1.00	0.93	0.96

Table 1

For normal emails

	Precision	Recall	F1-measure
BNB	0.99	1.00	1.00
SVM	1.00	1.00	1.00
MLP	1.00	1.00	1.00
DT	1.00	1.00	1.00
KNN	0.99	1.00	0.99

Table 2

The evaluation is based on the following metrics:

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

$$Recall = \frac{TP}{TP + FN}$$

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

where TP, FP and FN are the numbers of true positive, false positive and false negative results respectively. Table 1 and Table 2 shows the precision, recall, F-measure and time cost for training and testing by different algorithms. After discussion, we think that the reason for the relative low measure-score for spam is that spam emails are not always the same, they have more types than normal ones, which means that they can be spam for many reasons like virus links, ad calls, meaningless contents. We are happy to see that the classification of normal mail is very accurate, because classifying normal mail as spam can be disastrous, it will cause a more terrible loss than wasting time opening a spam, it may make us miss something important, and it is much worse than identifying spam as normal ones.

Figure IV-1 shows the screenshot of part of the result of the training program,

```
The train of BNB classifier is done. It cost 3.3104591369628906 s
 Here is the report:
              precision    recall  f1-score   support

           0       0.99      0.94      0.97       298
           1       0.99      1.00      1.00      1922

    accuracy                           0.99      2220
   macro avg       0.99      0.97      0.98      2220
weighted avg       0.99      0.99      0.99      2220
```

Figure IV-1

We also wrote a test program which can load the trained classifiers and do predictions about emails, Figure IV-2 shows a result of the test program,

```
Please enter the path of email you want to test: useless/ham1.txt
It's a normal email!
```

Figure IV-2

V. Comparison and Discussion

Running Speed

We also write code to see the time cost by the classifiers, the result is shown below:

	Time/s
BNB	1.52
SVM	46.2

MLP	99.8
DT	14.3
KNN	237

Obviously, Bernoulli NB is the quickest and knn is the slowest, after discussion, we find that Naïve Bayes is the best and simplest algorithm for simple text classification, also, the features we created were all sparse multivariate discrete values and Bernoulli NB can do better. For high dimensional features, For high-dimensional data processing, it will be difficult for knn to deal with, there will be "dimensional disaster" problems, and slow filtering speed

Possible Improvement

The spam classifier we make is only a foundation, where a lot of things are idealized, the e-mail will be much more complicated in real life, they often contain some accessories such as virus, rather than just advertising links or sell phone, the dictionary we created also has defects, the dictionary includes all words appearing in the training set, without considering the frequency, which can cause some impact on the result.

Conclusion

During this project, we allocate the tasks and cooperate to finish it. Everyone worked hard to get it done. Besides, we have learnt something new like classification algorithm we didn't learn in class. Moreover, our goal is reached, the accuracy, precision and recall are higher than 93%. We enjoyed this project and we hope our achievement can do other people a favor.