Project 5 - Problem 1 Spam Filtering Using Naïve Bayes

Submitted by:

Weiliang Xing (SBU ID: 108211104)
Md Atiqur Rahman (SBU ID: 109940054)

Introduction

Email junk problems are serious in nowadays. To reduce the negative effect of junk emails on user's email management, several strategies were developed to filter junk emails while hold the legitimate emails like manually setting the filter rules. However, they are limited and problematic with using large amount of time to build and to maintain rules, which may also be error-prone. The solution in the publication [3] we will implement is to use Naïve Bayes Classifier to filter junk emails based on Bayesian Network. A Bayesian classifier is a Bayesian network applied to a classification task. There are various types of Bayesian classifier based on different assumption, among which the Naïve Bayesian classifier is the oldest and most restrictive, which assumes that each feature is conditionally independent of every other features, given the class variable. We will implement the Naïve Bayesian classifier based on the procedure and mechanism shown in the publication.

Experiments:

Implementation procedure is shown below as the publication [3] claimed:

1) Import and preprocess the data from train.txt;
2) Eliminate all features that appear fewer than three times;
3) Calculate the mutual information, which is inspired by following formula:

$$I(U;C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U=e_t, C=e_c) \log_2 \frac{P(U=e_t, C=e_c)}{P(U=e_t)P(C=e_c)},$$

$$\begin{aligned} I(U;C) = {} & \frac{N_{11}}{N}\log_2\frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N}\log_2\frac{NN_{01}}{N_{0.}N_{.1}} \\ & + \frac{N_{10}}{N}\log_2\frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N}\log_2\frac{NN_{00}}{N_{0.}N_{.0}} \end{aligned}$$

Where U is a random variable that takes values $e_t = 1$ (the email contains feature t) and $e_t = 0$ (the email does not contain t), and C is a random variable that takes values $e_C = 1$ (the email is in class c) and $e_C = 0$ (the email is not in class c). Ns are counts of features appearing in the email entity. For example, N_{10} is the number of the emails that contain t ($e_t = 1$) and are not in c ($e_C = 0$)[1].

4) Select the first 500 features whose MI value is greatest to build the classifier;

5) We will use Multinomial Naïve Bayes to build the classifier by two steps shown below:

Calculate $P(c_j)$ terms
- For each c_j in C do
  $docs_j \leftarrow$ all docs with class $= c_j$

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- Calculate $P(w_k \mid c_j)$ terms
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word w_k in *Vocabulary*
    $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

$$P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

6)

where, docs here is the emails, P(cj) is the probability of class j, vocabulary is the collection of all features, α is the smoothing parameter which is used to adjust the result to avoid probability that may equal to 0 which made whole process failed, and wk is the specific feature/word in the vocabulary[2]. Here we will use Laplace smoothing[4] to set the smooth paremeterαto smooth categorical data.

7) According to the publication [2], we will use multinomial naïve Bayes classifier to do learning:

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c_j) \prod_{x \in X} P(x \mid c)$$

8) After building the classifier, we use it for each email in test group from test.txt. Due to potential underflow issues, we make each test item logarithmic and compare between the results from different classes and choose the class, which has larger value.
9) Recording the results and comparing the results with the truth in the test set to calculate the statistics.

Results and Conclusion:

Figure1 shows the results, where total precision is the precision of both junk precision and legitimate precision; junk precision is the precision that junk is correctly identified, and legitimate precision is the precision that legitimate is correctly identified. They are listed with different smoothing parameter α. According to the publication [3], the cost for misclassifying a legitimate E-mail as junk far outweighs the cost of marking a junk as legitimate. Thus to find the best accuracy with smoothing parameter α, we choose junk precision, which is of greatest concern to most users. The result also shown in Table 1.

From the Table1, we found that range of α from 6000 to 7000 will have highest junk precision, which is 98.45%; while in the range of α within 200, the highest total precision is 90.4% and the highest legimate precision is about 82.0% in the same range.

Recall the results from the publication[3], the highest precision in junk precision is 97.1% while legitmate is 87.7%, which are comparable with the results from our implementation.

By realizing the implementation for the publication[3], we reproduced the results shown in the publication, which confirmed the correctness of the implementation here. From this project, we learned different algorithms and core procedures to releaze Bayes classifier in practical problems, also the importance of smooth parameters on the results.

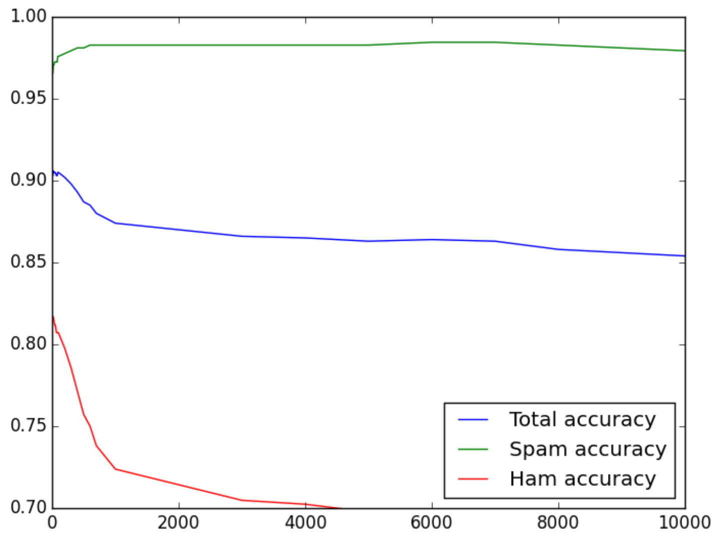

Figure 1. Result figure

Alpha	Total precision	Junk precision	Legitimate precision
0.1	0.904	0.965517241	0.819047619
0.2	0.904	0.965517241	0.819047619
0.3	0.904	0.965517241	0.819047619
0.4	0.904	0.965517241	0.819047619
0.5	0.904	0.965517241	0.819047619
0.6	0.904	0.965517241	0.819047619
0.7	0.904	0.965517241	0.819047619
0.8	0.904	0.965517241	0.819047619
0.9	0.904	0.965517241	0.819047619
1	0.904	0.965517241	0.819047619
10	0.903	0.965517241	0.816666667
20	0.906	0.970689655	0.816666667
30	0.905	0.970689655	0.814285714
40	0.905	0.972413793	0.811904762

50	0.905	0.972413793	0.811904762
60	0.904	0.972413793	0.80952381
70	0.903	0.972413793	0.807142857
80	0.903	0.972413793	0.807142857
90	0.905	0.975862069	0.807142857
100	0.905	0.975862069	0.807142857
200	0.902	0.977586207	0.797619048
300	0.898	0.979310345	0.785714286
400	0.893	0.981034483	0.771428571
500	0.887	0.981034483	0.757142857
600	0.885	0.982758621	0.75
700	0.88	0.982758621	0.738095238
800	0.878	0.982758621	0.733333333
900	0.876	0.982758621	0.728571429
1000	0.874	0.982758621	0.723809524
2000	0.87	0.982758621	0.714285714
3000	0.866	0.982758621	0.704761905
4000	0.865	0.982758621	0.702380952
5000	0.863	0.982758621	0.697619048
6000	0.864	0.984482759	0.697619048
7000	0.863	0.984482759	0.695238095
8000	0.858	0.982758621	0.685714286
9000	0.856	0.981034483	0.683333333
10000	0.854	0.979310345	0.680952381
20000	0.848	0.974137931	0.673809524
30000	0.837	0.965517241	0.65952381

40000	0.832	0.960344828	0.654761905
50000	0.824	0.951724138	0.647619048
60000	0.818	0.948275862	0.638095238
70000	0.809	0.944827586	0.621428571
80000	0.799	0.943103448	0.6
90000	0.795	0.94137931	0.592857143
100000	0.786	0.94137931	0.571428571
200000	0.731	0.939655172	0.442857143
300000	0.693	0.94137931	0.35
400000	0.675	0.948275862	0.297619048
500000	0.654	0.948275862	0.247619048
600000	0.643	0.955172414	0.211904762
700000	0.639	0.962068966	0.192857143
800000	0.634	0.963793103	0.178571429
900000	0.633	0.968965517	0.169047619
1000000	0.625	0.972413793	0.145238095

Table 1. Result Table

References:

[1] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze.*Introduction to information retrieval*. Vol. 1. Cambridge: Cambridge university press, 2008.
[2] Dan Jurafsky, Text Classification and Naïve Bayes. Stanford University.
[3] Sahami, Mehran, et al. "A Bayesian approach to filtering junk e-mail." *Learning for Text Categorization: Papers from the 1998 workshop*. Vol. 62. 1998.
[4] http://en.wikipedia.org/wiki/Additive_smoothing.