🡺 Topic

🡪 Spam Filter

Discovery Challenge <http://www.ecmlpkdd2006.org/challenge.html> -- A competition deals with personalized spam filtering and generalization across related learning tasks.

🡺 Content

-- Data Structure

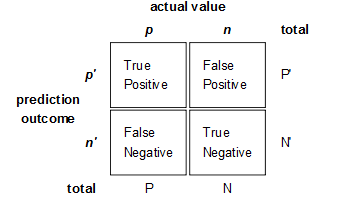
Give an example, the first line in task\_a\_labeled\_train.tf starts like this:  
1 9:3 94:1 109:1 163:1  
This line represents a *spam* email (starting with class label "1") with four words. The word ID of the first token is 9 and the word occurs 3 times within this email, indicated by ":3".

-- Train, Tune

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Task* | *File name* | *Data set size* | *Description* | ***Usage*** |
| Task A | task\_a\_labeled\_train.tf | 4000 emails | Labeled training emails. | **train** |
|  | task\_a\_u00\_eval.tf ... task\_a\_u02\_eval.tf | 2500 emails each | Unlabeled evaluation data: 3 inboxes from different users. | **evaluate, get the average AUC** |
|  | task\_a\_labeled\_tune.tf | 4000 emails | Labeled training emails for parameter tuning. Feature representation corresponds only to file task\_a\_u00\_tune.tf. | **build model,**  **tune parameter for u01 & u02** |
|  | task\_a\_u00\_tune.tf | 2500 emails | Labeled test emails of one user's inbox for parameter tuning. Feature representation corresponds only to file task\_a\_labeled\_tune.tf. | **tune parameter for u00** |

- Accuracy, ROC

The program for the calculation of the AUC is written in C and compiles with gcc.

[](http://bubblexc.com/wp-content/uploads/2011/03/Confusion-matrix2.png)

ROC (Receiver Operating Characteristic)

X axis -- Specificity -- False Positive rate = FP / [FP + TN]

Y axis -- Sensitivity -- True Positive rate = TP / [TP + FN]

🡺 Two methods

Bayesian Filtering & Logistic Regression

Also applies some optimized methods, like TF-IDF, anti-spoof, self-learning, etc.

🡺 Machine Learning Techniques Types

-- Discriminative Model (Conditional Model), e.g. Logistic Regression, SVM

By modeling the conditional probability distribution P(y|x), which can be used for predicting y from x.

-- Generative Model, e.g. Naive Bayes

Specifies a joint probability distribution over observation and label sequences.

🡺 Bayesian Filtering

A PLAN FOR SPAM <http://www.paulgraham.com/spam.html> , Paul Graham

-- Concept (Bayesian Combine)

Combining Probabilities <http://www.mathpages.com/home/kmath267.htm>

Given a evaluate mail, we can calculate

P =

, where is the probability of a mail to be Spam when it contains the word W1

Assume P(S) = 0.5, then P =

So, we just need to know P(S|W) for each Word.

P(S|W) = , we can also assume P(S) = P(H) = 50%

, where P(W|S) is the static likelihood of a word exists in all Spam, which we call learned from training data. Same as P(W|H) .

-- Procedure

Implement Graham’s method.

1. Learn the likelihood of each word existing in Spam and Nonspam from train\_data or 1st-half of tune\_data;

e.g. There are 2000 Spam and 2000 Nonspam in train\_data, the number of Spam which has word W1 is 1234, the number of Nonspam which has word W1 is 567. So, P(W|S) = 1234/2000, P(W|H) = 567/2000;

2. We can now evaluate each mail from u01\_eval, u02\_eval, or the 2nd-half of tune\_data;

For each word in a mail, we calculate its P(S|W), and then we sort all the P(S|W) of words in this mail by Decreasing order. And we just pick the first some of these words (e.g. 15, called **interesting words**) to calculate the Combining Probability of this mail. This is the probability of this mail to be a Spam.

e.g. P = P1P2…P15 / P1P2…P15 + (1-P1)(1-P2)…(1-P15), where P1 is the simple symbol of P(S|W1)

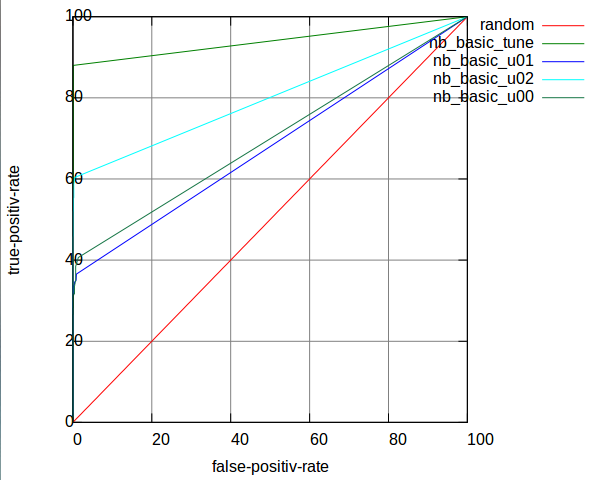
We can predict whether a mail is a Spam or not based on this probability P.

3. There are some parameters not decided yet for evaluation. So the tune\_data is used to **decide** those paramters. We cannot do this tuning process directly on u02\_eval because if we did so, we would cheat based on test data. And, we cannot add tune\_data into train\_data because the word ID of these two files is different.

4. For u00, we have also a sample of u00\_tune\_data, which could be used to tune parameters only for u00\_eval. With this different parameters, we also train on train\_data and then evaluate on u00\_eval.

-- Basic Result (tune, u00, u01, u02; u00-tune, u00-tuned; ROC)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | tune | u01 | u02 | u00\_tune | u00 |
| AUC | 0.998056 | **0.899351** | **0.951194** | 0.992355 | 0.869827 |
| u00\_tuned |  |  |  | 0.996812 | **0.894598** |



-- Improvement (frequency, and anti-spoof)

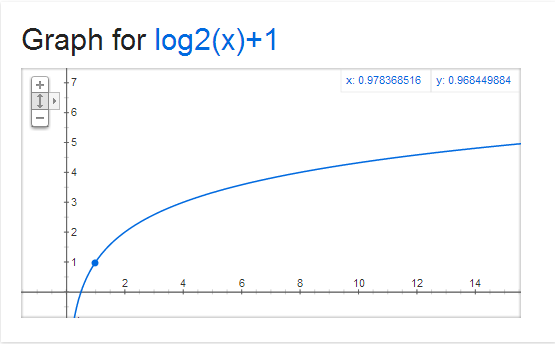
When I statistic the likelihood, I only +1 time when a word exists in Spam or Nonspam. I have not taken advantage of the frequency information.

But if I simply add the true frequency value, it would also not reflect the real likelihood.

e.g. in train\_data, -1 9:84 35:324 67:2 74:18 92:6 132:3

The useful frequence should distinguish the low value, but should not be too big when the value is high.

I hope the frequence can be transformed to something like this figure.



However, there is another problem pointed out by Graham, “such an algorithm would be **easy** for spammers **to spoof**: just add a big chunk of random text to **counterbalance** the spam terms.”

So, I will omit words having too high frequence only in Spam, not in Healthy mails. Because the Nonspam mails rarely use too many same words.

Here are my optimized functions:

if label == 1 {

if word.split(":").last.to\_i >= @min\_freq {

freq = Math.log(word.split(":").last.to\_i,2).to\_i + 1

if freq > @max\_freq {

freq = 0

else

@spam[@index] += freq

}

}

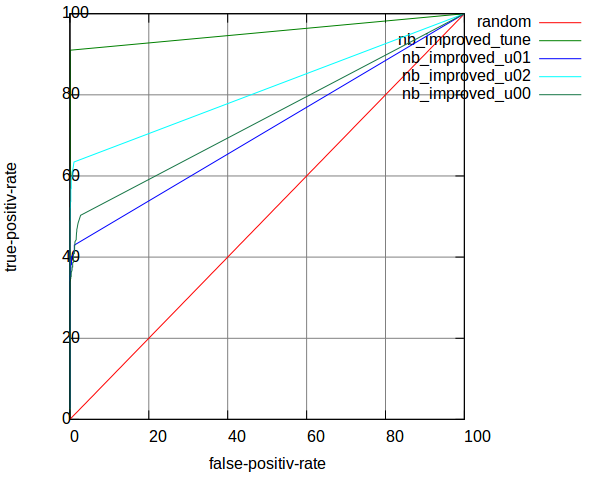
else if label == -1

@nonspam[@index] += freq

}

-- Optimized Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | tune | u01 | u02 | u00\_tune | u00 |
| AUC | 0.998196 | **0.912002** | **0.952196** | 0.991569 | 0.874134 |
| u00\_tuned |  |  |  | 0.991257 | **0.901313** |



🡺 Logistic Regression

Online Discriminative Spam Filter Training, Joshua Goodman & Wen-tau Yih, Microsoft Research

-- Concept (formula)

We will learn a set of weights for each word in the body or header of the message. When a new message arrives, we find this list of words, and sum the weights associated with those words.

, where is vector of weights, and is a vector of 1’s or 0’s with a 1 in the position corresponding to each word.

The equation results a probability between 0 and 1. If the probability is over some threshold, we predict that the email is spam; otherwise predict the email is ham.

-- Procedure

Given algorithm below:

// initialize weights to 0

For each ,

if (p>0.5)

predict spam;

else

predict ham;

if( == 1 )

else

Suppose threshold is 0.5 here; learning rate makes sure that the step taken in the direction of the gradient is not too large

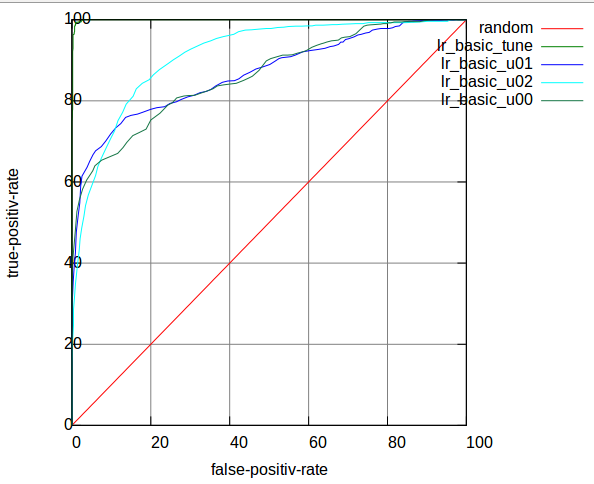
1. Learn on train\_data;

2. Evaluate each mail based on , so that we could get probability of a mail being Spam given words vector of this mail;

3. The parameter tuning and train to evaluate process are similar to that of Bayesian Filtering.

-- Basic Result (tune, u00, u01, u02; u00-tune, u00-tuned; ROC)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | tune | u01 | u02 | u00\_tune | u00 |
| AUC | 0.999212 | **0.867380** | **0.912238** | 0.999411 | 0.848224 |
| u00\_tuned |  |  |  | 0.999588 | **0.862130** |



-- Improvement (TF-IDF failed, M-TF-IDF(improved TF-IDF,M-TF-IDF), Modified-TF-IDF)

**First attempt:**

Logistic Regression with TF-IDF is a common technique in text classification. TF-IDF is generally used in search engine. It is a numerical statistic which reflects how important a word is to a document in a collection.

We use TF-IDF here to restructure the weight vector of each word.

-- term frequency = frequency of a term t in document d / total words count (or most frequent words count) in document d

-- inverse document frequency =

TF-IDF = , it can evaluate how important the word is in Spam/Nonspam.

=

, where Wi is the vector of weights, is term frequency of word i in the e-mail, M is the total number of the email collection, N is the total number of the words in a detailed email.

It also takes the importance of each word into consideration onto weight, which is similar to the method used for picking most featured words in Bayesian Filter.

**Second attempt:**

Improved Feature Selection Algorithm in Spam Filtering Based on TF\*IDF, by Chen Qi, Wu Zhao-hui, etc. @Yanshan University, China

Limitation of TF-IDF:

If a word has high term frequency (TF) in both Spam and Nonspam, it does not has good discriminant validity. This also happens in inverse document frequency (IDF).

Improved TF-IDF:

, where is the ith word term frequency in Spam, and is the ith word term frequency in Healthy. S is total number of Spam, H is total number of Healthy.

By evaluating the difference of word’s TF and IDF between Spam and Nonspam, improved TF-IDF can better show the word’s discriminate validity, especially the low frequency words.

But these two attempts didn’t work well, it even decreased the AUC in u02.

|  |  |  |  |
| --- | --- | --- | --- |
|  | u00 | u01 | u02 |
| AUC | 0.859492 | 0.867861 | 0.847229 |

The second attempt sound reasonable, it should have taken effect. But it has **missed some important concept in original TF-IDF**. It has not taken the evaluate mail itself into consideration.

Simple =

Improved =

Further improved = , where I added the denominator here, to taken the feature of a evaluate mail into consideration. This increased the AUC decently.

id = 0

\_tfidf\_.each do |key, value|

if id < @pick\_limit

w\_x += @weight[key] \* value / max\_frequency

end

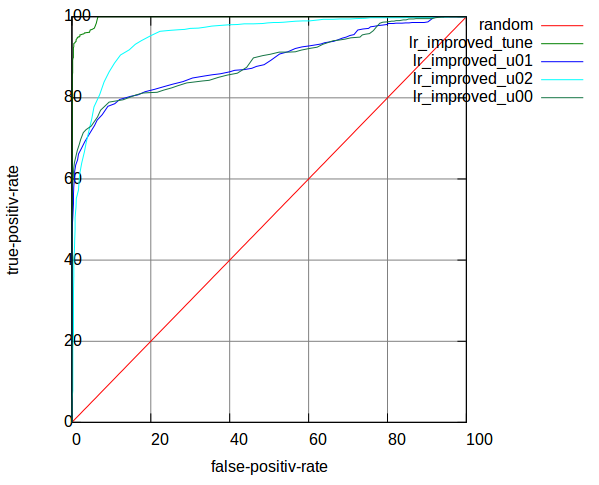
id += 1

end

, where \_tfidf is sorted by DESC

-- Optimized Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | tune | u01 | u02 | u00\_tune | u00 |
| AUC | 0.999371 | **0.887771** | **0.952392** | 0.999475 | 0.881424 |
| u00\_tuned |  |  |  | 0.999606 | **0.888191** |



🡺 Shall we stop?

So we get [0.901313, 0.912002, 0.952196] on Bayesian Filter, [0.888191, 0.887771, 0.952392] on Logistic Regression Filter. Shall we stop?

**No.**

Observe: why AUC on tune\_data could be so high? Why u02 has higher AUC than u01, and u00 has lowest AUC but has an individual tuning data?

Because the features of these 3 inboxes are different. If the features between training data and evaluation data are similar, then I guess we can as high accuracy as what we get on divided tune\_data, whose 1st-half partition is used for training, and the 2nd-half partition is used for evaluating.

-- I can learn from myself

-- More data, more good.

-- Avoidcheating

I cannot learn directed from truly labeled testing data which provided for evaluation. But I can learn from my own evaluation result. That means, I should have confidence that my Spam Filter is good enough. I call this self-learning, and I guess this method is used in practical, like gmail.

-- Improvement

Because I don’t know the true label when evaluating mails, I should not learn all of them. But I can still learn some part of those evaluating data based on the confidence I have on them.

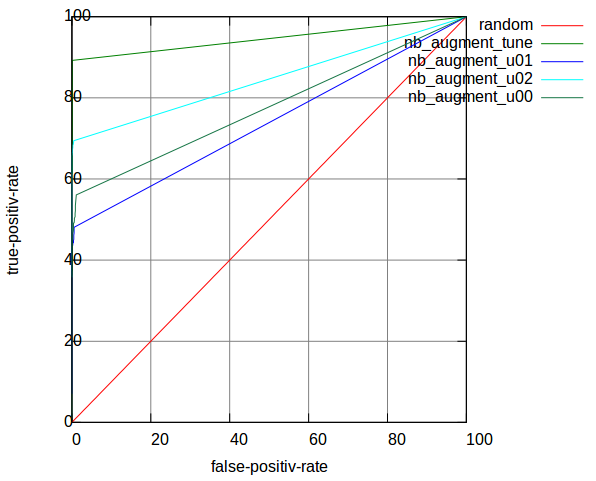
So, I set a learning boundary when evaluating mails. I believe those mails which have probabilities > 0.99 or < 0.01 are predicted correctly. Then I can treat those mails as labeled to **augment** my training data.

Another advantage is that these extra training data has more **similarities** on features to the evaluating data.

-- Optimized result

Bayesian Filter

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | tune | u01 | u02 | u00\_tune | u00 |
| AUC | 0.997701 | **0.927123** | **0.980120** | 0.989169 | 0.889985 |
| u00\_tuned |  |  |  | 0.992031 | **0.958331** |



Logistic Regression Filter

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | tune | u01 | u02 | u00\_tune | u00 |
| AUC | 0.999315 | **0.899148** | **0.960306** |  |  |
| u00\_tuned |  |  |  | 0.999421 | **0.892779** |

The Bayesian Filter has been benefited a lot from self-learning, but Logistic Regression Filter didn’t. We will analyze the reason afterwards. Hope it is true. The key word is converge speed of LR.

-- Shall we stop again? It depends.

Since the self-learned data has more similarities to those evaluation data, we want to increasing their weight while evaluation. We can import an enlargement coefficient on the self-learning rate. I got this idea from LR learning rate.

But this enlargement is obviously not the bigger the better. When we learned from predicted probability, we also unavoidably learned some bad classified mails. That’s not good for this kind of mistakes to be enlarged.

Bayesian Filter

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Enlarged x3 | |  |  |  |  |
|  | tune | u01 | u02 | u00\_tune | u00 |
| AUC | 0.997126 | **0.936075** | **0.985456** |  |  |
| u00\_tuned |  |  |  | 0.990459 | **0.976759** |
| Enlarged x20 | |  |  |  |  |
|  | tune | u01 | u02 | u00\_tune | u00 |
| AUC | 0.994268 | **0.953053** | **0.987347** |  |  |
| u00\_tuned |  |  |  | 0.979197 | **0.988825** |

🡺 Conclusion

-- LR

Weights will trend to a balanced point, converge faster

Need limited train data compared to Bayes

Not improved by self-learning

-- Bayes

More data more precise

Good performance when self-learning

“Spam filtering is not just classification, because false positives are so much worse than false negatives that you should treat them as a different kind of error. And the source of error is not just random variation, but a live human spammer working actively to defeat your filter.”

-- Paul Graham

-- My Results

**Optimized NB & LR**

Bayesian Filter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | u00 | u01 | u02 | Average AUC |
| AUC | 0.901313 | 0.912002 | 0.952196 | 0.921837 |

Logistic Regression Filter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | u00 | u01 | u02 | Average AUC |
| AUC | 0.888191 | 0.887771 | 0.952392 | 0.909551 |

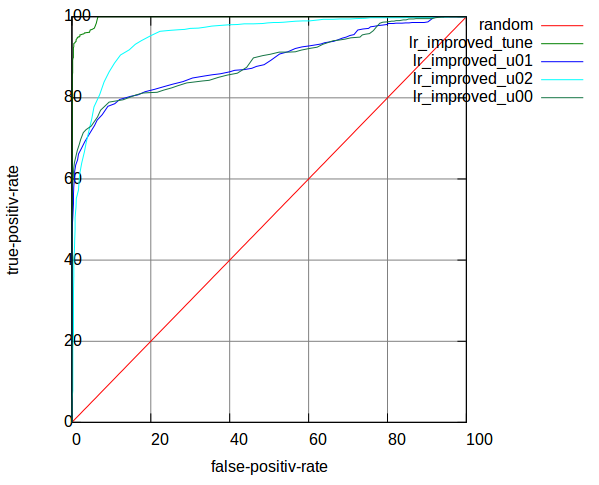
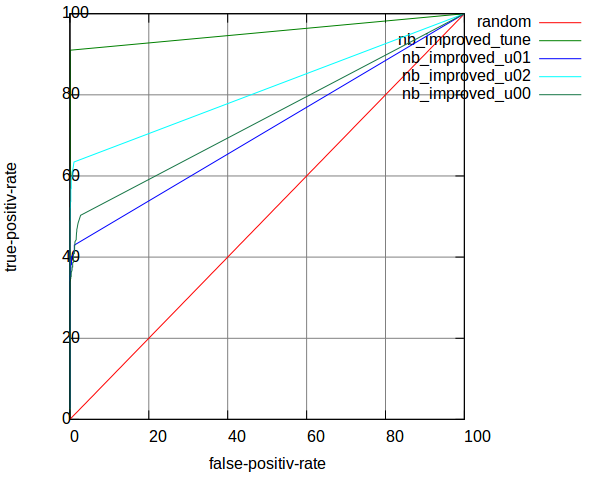
🡪With self-learning

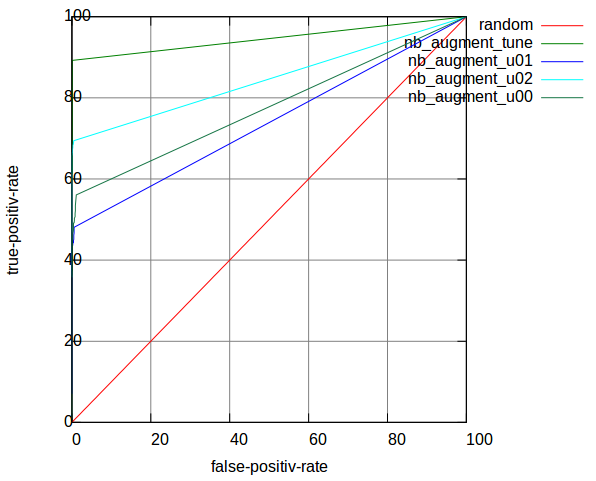
Bayesian Filter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | u00 | u01 | u02 | Average AUC |
| AUC | 0.958331 | 0.927123 | 0.980120 | 0.955191 |

Enlarge x3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | u00 | u01 | u02 | Average AUC |
| AUC | 0.976759 | 0.936075 | 0.985456 | 0.966097 |





I feel quite exciting when compared to Ranking in Discovery Challenge 2006:

|  |  |  |
| --- | --- | --- |
| *Rank Task A* | *Average AUC* | *Team* |
| 1 | 0.950667 | **Khurram Nazir Junejo, Mirza Muhammad Yousaf, Asim Karim** Lahore University of Management Sciences, Pakistan |
| 1 | 0.949094 | **Bernhard Pfahringer** University of Waikato, New Zealand |
| 1 | 0.948666 | **Kushagra Gupta, Vikrant Chaudhary, Nikhil Marwah, Chirag Taneja** Inductis India Pvt Ltd |
| 2 | 0.936457 | **Nikolaos Trogkanis**, National Technical University of Athens, Greece **Georgios Paliouras**, National Center of Scientific Research "Demokritos", Greece |
| 3 | 0.927839 | **Chao Xu, Yiming Zhou** Beihang University, Beijing, China |

Updated results Task A:

|  |  |
| --- | --- |
| *Average AUC* | *Team* |
| 0.9875 | **Khurram Nazir Junejo, Mirza Muhammad Yousaf, Asim Karim** Lahore University of Management Sciences, Pakistan |
| 0.9731 | **Antonia Kyriakopoulou, Theodore Kalamboukis** Athens University of Economics and Business |
| 0.9672 | **Alexander Zien (invited speaker),** [**∇S3VM algorithm**](http://www.kyb.mpg.de/publication.html?publ=4162) Max Planck Institute for Biological Cybernetics, Tuebingen, Germany |
| 0.9588 | **Nikolaos Trogkanis**, National Technical University of Athens, Greece **Georgios Paliouras**, National Center of Scientific Research "Demokritos", Greece |
| 0.9491 | **Bernhard Pfahringer** University of Waikato, New Zealand |
| 0.9487 | **Kushagra Gupta, Vikrant Chaudhary, Nikhil Marwah, Chirag Taneja** Inductis India Pvt Ltd |
| 0.9300 | **Gordon Cormack** University of Waterloo, Canada |

🡺 Further Improvement

-- Combine LR and NB

From the ROC curve, we can see the LR has relatively high fpr, as well as high tpr;

while NB has little fpr, but lower tpr.

I guess there maybe exist a way to combine those two methods, which will take advantage of them on different features.

e.g. I have tried to static the most false-positive/false-negative area in LR, which found out than the false-positive-error most occurred in probability ranges from [60, 80]. So, there is an idea that I can use LR as the 1st Filter, if its decision probability of a new mail is in range [60, 80], then I will use NB as the 2nd Filter there. After my attempt to do so, the AUC really increases, but too little to be caught in eyes.

-- Ada-boost

I’m sure Ada-boost will have effect on this kind of combination.