Email Classification with Naïve Bayes

Mitchell Della Marta, Shu Han Bor

May 4, 2014

1 Introduction

1.1 Aim

To implement text classification of emails using the Naïve Bayes classifier for spam detection.

1.2 Importance

As email is one of the main forms of communication, spam detection to remove spam is important. Email spam is an unsolved global issue that negatively affects productivity and uses up valuable resources. Spam emails are usually sent with malicious intent to users for some monetary gain. Email users spend a large amount of time regularly deleting these spam emails, which needlessly occupy storage space and consumes bandwidth. Hence the development of classifiers that are able to separate legitimate from spam emails is required. Classification of emails to separate and remove spam from other emails, aims to save the user from having to spend time and effort sorting through and deleting spam, and protect unaware users from malware.

Traditional non-machine learning based techniques are easily fooled by spammers and require periodic manual updates. Therefore, machine learning based methods are required, which are able to automatically analyse email contents and dynamically update themselves to cope with new spamming techniques.

2 Data Preprocessing

Before classification can be performed, we must represent the files in our sample set *LingSpam-mini600*[2] appropriately. Using the "bag-of-words" model, words are extracted from the file and treated as features. There are two main characteristics in an email; the subject and the body. Thus, we will construct two "bags-of-words", one for each component. To determine which corpus a feature will belong to, we if the line begins with "Subject:". If it does the words within the line will be added to the subject corpus, otherwise they will be added to the body corpus.

To each file, we performed the following steps:

1. Replace all punctuation and special symbols with a space.
2. Remove stop words, using a list of stop words "english.stop".
3. Remove numbers or words containing digits.

The tokenisation schemes used in the example data are different to that used in our stop words list. The data detaches clitics and contraction affices, whereas the stop words list is tokenised based on whitespace and punctuation characters. Therefore, we chose to replace punctuation with a space (e.g. "n't" becomes "n", "t") instead of simply removing them. This meant that the stop word list would remove clitics as it removes single characters, instead of keeping it as a feature as ("nt" isn't in the stop words list).

The *document frequency* feature selection method was then used to select the 200 words which occured in the most documents, and used to build a classifier for the given documents. The following steps were taken to accomplish this:

1. Counter is created for each email, which keeps track of the number of times each word appears in that email.
2. For each word that appeared in the counter for an email, it is added once to the counter for the corresponding subcorpus.
3. In each corpus the words with the top 200 document frequency score are selected as features to represent that corpus.

Note that different runs of our preprocessing script can result in different classifier accuracies. As the counter is stored in a hashmap, words with the same frequency before and after the 200 word cutoff are chosen arbitrarily. Depending on the words chosen, the accuracy of the classifier can be affected.

Each selected feature was then weighted using its *td-idf* score [1].

$$tfidf(t_k, d_j) = \#(t_k, d_j) \times \log \frac{|Tr|}{\#Tr(t_k)}$$

Then the scores were normalised, using *cosine normalisation* [1].

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|}(tfidf(t_s, d_j))^2}}$$

In cases where the *tf-idf* weightings for all 200 features for a given document is 0, the denominator of the cosine normalisation will also be 0. This is impossible, so instead we set w_{kj} to 0.

2.1 Data Characteristics

Table I: Characteristics of data set

	Subject	**Body**
# Features before removing stop words	1074	19886
# Features after removing stop words	915	19386
# $Class_{nonspam}$	600	600
# $Class_{spam}$	200	200

The “bag-of-words" model produced 19886 and 1074 features in the body and subject corpora respectively. After the removal of stop words, there were 19386 unique words remaining in the body corpus, and 915 words in the subject corpus (see Table I).

3 Feature Selection

Feature selection is performed using document frequency. This method involves computing the number of documents a word occurs in, for every word, and selecting the top 200 words with the highest scores to build a classifier.

Shown in Table II and Table III are the top 100 words for the subject and body corpora respectively and their document frequency score. Removing stop words filters out extremely common words that have little value in classification. It is beneficial in text processing, as it removes low quality features, allowing more significant features to have precedence. Most resulting words are sensible, although there are some aren’t (e.g. “qs”, “ll”, “ca”). However, removing these words results in a lower accuracy. Therefore, we have chosen to retain these as some of them may be help the classifier distinguish between spam and non-spam emails. Thus, given our task to process natural language text, the selection of words shown makes sense as it gives a better representation of the contents of the emails, and helps improve the accuracy of the classifier.

A comparison of Table II and Table III shows significant disparities in the document frequency of features and word distribution. The frequencies for the subject is significantly lower than that of the body. Whilst some features are shared between subject and body, most features selected are different.

Table II: Top 100 words in subject corpus and corresponding document frequency (DF) scores

Rank	Word	Score	Rank	Word	Score	Rank	Word	Score
1	sum	30	35	speaker	6	69	intuitions	4
2	summary	26	36	german	6	70	banning	4
3	english	24	37	internet	6	71	school	3
4	language	21	38	business	6	72	resolution	3
5	free	20	39	list	5	73	ary	3
6	disc	19	40	resources	5	74	adjectives	3
7	query	18	41	native	5	75	verbal	3
8	linguistics	15	42	research	5	76	teaching	3
9	comparative	13	43	word	5	77	future	3
10	sex	13	44	spanish	5	78	lists	3
11	opposites	12	45	linguist	5	79	background	3
12	words	12	46	jobs	5	80	synthetic	3
13	book	10	47	needed	5	81	credit	3
14	email	10	48	grammar	5	82	home	3
15	call	9	49	software	5	83	live	3
16	job	9	50	languages	5	84	youthese	3
17	method	9	51	time	5	85	uniformitarianism	3
18	japanese	8	52	fwd	4	86	released	3
19	correction	8	53	summer	4	87	names	3
20	syntax	7	54	address	4	88	opportunity	3
21	program	7	55	books	4	89	decimal	3
22	qs	7	56	information	4	90	world	3
23	chinese	7	57	request	4	91	misc	3
24	announcement	7	58	phonetics	4	92	sites	3
25	million	7	59	pig	4	93	double	3
26	part	6	60	american	4	94	acquisition	3
27	slip	6	61	programs	4	95	site	3
28	workshop	6	62	unlimited	4	96	policy	3
29	armey	6	63	web	4	97	fall	3
30	money	6	64	www	4	98	teach	3
31	lang	6	65	secrets	4	99	hey	3
32	conference	6	66	great	4	100	line	3
33	dick	6	67	read	4			
34	mail	6	68	systems	4			

Table III: Top 100 words in body corpus and corresponding document frequency (DF) scores

Rank	Word	Score	Rank	Word	Score	Rank	Word	Score
1	information	205	35	message	91	69	full	74
2	language	192	36	ll	89	70	system	74
3	mail	183	37	receive	88	71	ac	73
4	university	179	38	check	88	72	today	73
5	time	178	39	phone	88	73	questions	72
6	list	171	40	good	87	74	remove	72
7	address	165	41	day	86	75	interest	72
8	english	159	42	interested	86	76	john	71
9	linguistics	156	43	year	86	77	found	70
10	http	156	44	include	85	78	related	70
11	people	146	45	working	85	79	site	69
12	send	146	46	case	85	80	linguist	69
13	free	144	47	based	84	81	usa	69
14	make	140	48	ve	84	82	text	68
15	email	133	49	note	83	83	point	68
16	number	128	50	home	83	84	read	68
17	work	128	51	made	83	85	ago	67
18	www	122	52	part	83	86	book	67
19	languages	119	53	including	81	87	week	67
20	find	118	54	mailing	81	88	making	66
21	fax	116	55	type	80	89	dear	66
22	order	108	56	give	79	90	cost	66
23	call	103	57	program	79	91	question	65
24	form	101	58	web	79	92	simply	65
25	research	100	59	place	79	93	received	63
26	state	99	60	special	78	94	offer	63
27	linguistic	99	61	line	78	95	general	63
28	subject	98	62	date	78	96	important	62
29	years	98	63	days	77	97	data	62
30	world	98	64	back	76	98	ca	61
31	contact	97	65	internet	76	99	long	61
32	de	96	66	american	75	100	summary	61
33	money	94	67	service	75			
34	word	91	68	business	74			

4 Subject vs Body Analysis

4.1 Results

We ran our preprocessed data against a variety of Weka classifiers; ZeroR, OneR, IBk with $k = 1$ (1-NN) and $k = 3$ (3-NN), NaiveBayes (NB), J48 (DT) and MultilayerPerceptron (MLP). We tested these with Weka's 10 fold cross-validation. We also ran it against our Naïve Bayes classifier, tested against our implementation of the 10 fold cross-validation.

Table IV: Various classifiers tested with 10 fold cross validation for both the subject and body corpora

	Accuracy (%)	
Classifier	**Subject**	**Body**
ZeroR	66.67	66.67
OneR	70.00	82.00
1-NN	79.00	87.17
3-NN	68.17	85.00
NB	68.50	94.83
DT	66.67	92.50
MLP	76.17	96.67
MyNB	80.83	95.33

Table V: Accuracy of classifiers from lowest (1) to highest (8) for both corpora

	Accuracy Ranking							
Corpus	1	2	3	4	5	6	7	8
Subject	ZeroR	DT	3-NN	NB	OneR	MLP	1-NN	MyNB
Body	ZeroR	OneR	3-NN	1-NN	DT	NB	MyNB	MLP

The comparison of the performance (accuracy) of MyNB and NB over the 10 folds was completed in the same execution as the respective results in Table IV. MyNB used our 10 fold cross-validation and NB used Weka's 10-fold cross-validation. The items in each fold (given to MyNB and NB) are expected to be different.

Table VI: Comparison of fold accuracy between our Naïve Bayes classifier (C_1) and Weka's (C_2) for the subject corpus

Fold Number	C1 (%)	C2 (%)	Difference
Fold 1	80.00	60.00	20.00
Fold 2	83.33	73.33	10.00
Fold 3	71.67	63.33	8.34
Fold 4	80.00	76.67	3.33
Fold 5	83.33	71.67	11.66
Fold 6	76.67	70.00	6.67
Fold 7	83.33	73.33	10.00
Fold 8	86.67	63.33	23.34
Fold 9	81.67	73.33	8.34
Fold 10	81.67	60.00	21.67
Mean	80.83	68.50	12.34

Table VII: Comparison of fold accuracy between our Naïve Bayes classifier (C_1) and Weka's (C_2) for the body corpus

Fold Number	C1 (%)	C2 (%)	Difference
Fold 1	98.33	98.33	0.00
Fold 2	98.33	93.33	5.00
Fold 3	96.67	95.00	1.67
Fold 4	90.00	100.00	10.00
Fold 5	91.67	95.00	3.33
Fold 6	96.67	93.33	3.33
Fold 7	93.33	95.00	1.67
Fold 8	95.00	91.67	3.33
Fold 9	96.67	91.67	5.00
Fold 10	96.67	95.00	1.67
Mean	95.33	94.83	3.50

4.2 Discussion

4.2.1 Comparison of Classifiers

The accuracy of a classifier in comparison to the other classifiers, varies slightly against the two corpora. Accuracy from lowest to highest for both corpora are shown in Table V. Although there

are some discrepancies, in general ZeroR, OneR, 3-NN, DT perform more poorly than 1-NN, NB, MyNB, MLP.

The ZeroR classifier performs poorly for both corpora. It returns the majority class given from the training data. As a result ZeroR returns "nonspam" as there are 400 non-spam and 200 spam cases. Thus ZeroR produces a correct answer for and only for test data of the majority class, "nonspam", which is

$$\begin{aligned}\frac{\#nonspam}{\#documents} &= \frac{360}{540} \\ &= 66.67\%\end{aligned}$$

as obtained.

OneR works similarly to ZeroR, but instead of having no rules, it generates one rule for each item in the data set. After calculating rules for each, it will select the rule that gives the smallest total error. Although its accuracy isn't high, it is decent for the simplicity and speed of the algorithm.

In contrast to these two classifiers, multilayer perceptron (MLP) is more complex and returns the highest accuracy for body, and the second highest accuracy for subject. MLP is defined as "consists of multiple layers of simple, two-state, sigmoid processing elements (nodes) or neurons that interact using weighted connections." (Pal 1992, p.684). The Weka MLP assigns the features as output neurons with a random weight. Weka's default number of hidden neurons[3] was used,

$$\begin{aligned}\frac{(attributes + classes)}{2} &= \frac{(200 + 2)}{2} \\ &= 101\end{aligned}$$

As the network is then trained using back propagation, which corrects the weights to minimize the error in the entire output, MLP has a high accuracy. However, although it is very accurate, MLP takes significantly more time to train than the other classifiers.

Naïve Bayes (NB) is based on the Bayes rule of conditional probability. It analyses each attribute individually, by assuming they are all of equal importance and independent of one another. Although it is based on a simple concept, our MyNB classifier outperforms MLP which is a comparatively more complex algorithm for the subject corpus, and performs almost as well as MLP for the body corpus.

Whilst NB and MyNB are based on the same principle, MyNB outperforms NB. The difference in performance is due to implementation. The major difference between implementations is that NB uses numeric estimator precision values that are chosen by calculating the differences between adjacent values. This allows them to form a kernel density estimation, that estimates a probability density function (PDF), as opposed to using a normal PDF. On the other hand, we simply assume the data has a normal probability distribution function.

1-NN and 3-NN are both types of the k-nearest neighbour algorithm, where $k = 1$ and $k = 3$ respectively. Although they use similar algorithms, 1-NN performs better than 3-NN. 1-NN takes one neighbour, whereas 3-NN takes three. Therefore, this disparity is probably because in most cases the closest neighbour correctly classifies the new example, but the next two closest

do not. Hence 1-NN gets a greater accuracy than 3-NN. These classifiers work moderately well compared to the other classifiers, as each class is characterised by a large number of features, which the nearest neighbour algorithm takes advantage of. In order to increase the accuracy of this algorithm, it should be given a larger training set. Although the benefits of this have to be weighed against the time it takes to find the nearest neighbour in a larger training set.

Weka uses J48 (implementation of Quinlan's C4.5) to construct the decision tree classifier, which relies on using information gain to select attributes that best classifies the data. It performs averagely on the body corpus, and relatively poorly on the subject corpus.

4.2.2 Comparison of myNB vs Weka's NB

We conduct a paired t-test in our comparison to determine whether the differences in accuracy between our and Weka's Naïve Bayes was statistically significant, with a confidence level of 95%. To carry out the paired t-test:

1. Calculate the difference between the two classifiers in each fold

$$|c_{1_i} - c_{2_i}| \text{ for } i \in [1, \ldots, N]$$

   where $c_{1_i} \in C_1$ and $c_{2_i} \in C_2$ for all i.
2. Calculate the sample standard deviation of the differences:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

   where $X = \{x_1, \ldots, x_N\}$ are the observed differences.
3. Calculate the $(1-\alpha)$-upper confidence interval (UCL) of the mean:

$$\text{UCL}_{1-\alpha} = \bar{X} \pm t_{(1-\alpha)(k-1)} s_{N-1}$$

   for the confidence level $(1-\alpha) = 0.95$.
4. If the interval $\text{UCL}_{0.95}$ contains 0 then the difference in accuracy is not statistically significant; otherwise, it is.

Using the differences calculated in Table VI for the subject corpus we find that:

$$\begin{aligned} s &= 6.86 \\ \text{UCL}_{0.95} &= \bar{X} \pm t_{0.95,9} \cdot s_{N-1} \\ &= 12.34 \pm 2.262 \times 6.86 \\ &= 12.34 \pm 15.52 \\ &= [-3.18, 27.85] \end{aligned}$$

Similarly, given the differences calculated in Table VII, we obtain for body corpus:

$$\begin{aligned} s &= 2.77 \\ \text{UCL}_{0.95} &= \bar{X} \pm t_{0.95,9} \cdot s_{N-1} \\ &= 3.50 \pm 2.262 \times 2.77 \\ &= 3.50 \pm 6.27 \\ &= [-2.77, 9.77] \end{aligned}$$

For the subject corpus $\text{UCL}_{0.95} = [-3.18, 27.85]$ and for body corpus $\text{UCL}_{0.95} = [-2.77, 9.77]$. In both cases the interval $\text{UCL}_{0.95}$ contains 0. Therefore, the difference between the two Naïve Bayes classifiers for both subject and body are not statistically significant.

4.2.3 Comparison Between Subject and Body Corpora

The overall performance of the classifiers regardless of type, perform significantly better on the body than the subject corpus. This suggests that the email body is a better indicator of content.

5 Challenge Analysis

5.1 Results

5.2 Discussion

6 Conclusion and Future Work

In this report, we apply Naïve Bayes to classify emails after applying document frequency for feature selection. Our classifier is relatively accurate compared to a variety of other classifiers with an accuracy of 95.33% on body, and 80.83% on subject. We also examined a variety of feature selection methods and their effect on accuracy. We found that information gain gave us the highest accuracy with 97.5% on the body corpus.

Therefore, based on our experimental results, Naïve Bayes is well suited for email classification to detect spam, especially when paired with information gain.

Our future research work will deal with creating more meaningful features. This includes implementing stemming and creating tokens using more than one word. Currently, the tokens used in our classifier are formed from single words. Therefore, it will not analyse common consecutive words that are found in spam emails. By taking into account permutations of consecutive words, or words that appear within a specified distance of each other, the accuracy of our Bayesian classifier could be increased.

7 Reflection

8 Instructions

8.1 Prerequisites

- Either python3.3 or python3.4 installed.
- *lingspam-mini600* data set placed in the working directory (i.e. same directory as *process.py* and *naivebayes.py*).

8.2 Running Scripts

1. Use `python process.py` to run preprocessing script to create *body.csv* and *subject.csv*. These files are then accessible in the same directory.
2. Use `python naivebayes.py` to run Naïve Bayes classifier on the csv files. Result of accuracy calculation on both subject and body will be output.

You can customise the preprocessing by changing the function used to create `body_features` and `sub_features` in *process.py*.

References

[1] Fabrizio Sebastiani, *Machine learning in automated text categorization.* ACM Computing Surveys, 34(1):1-47, 2002.

[2] LingSpam has been collected by Ion Androutsopoulos and is described in the following paper: "An Evaluation of Naive Bayesian Anti-Spam Filtering" by I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, George Paliouras, and C.D. Spyropoulos; *In Proceeding of Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning*, Barcelona, Spain, 2000.

[3] M. Ware. n.d. (2014, April 30) *Class MultilayerPerceptron* (Revision 10169) [Online]. Available: http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html