TEXT BASED SPAM FILTER AND FEATURE WEIGHING

Sushant Pritmani pritmani.s@husky.neu.edu

Koosh Doctor doctor.k@husky.neu.edu

ABSTRACT

Unsolicited commercial email or spam have been continuously a pressing problem of the Internet. While many of us have become trained in realizing when an email is spam, there are still a lot of us who fall for money scams, dubious pharmaceutical products, malware, phishing websites, fraudulent products and services. On the other hand, the spam emails are undermining the usability of the email system and costing businesses heavily. Unfortunately, the businesses are not winning in legal aspects of spam, hence they have to resort to spam filters. As history certifies Naïve Bayes Classification as a popular method for text classification, we have used this family of probabilistic classifier along with n-gram word frequencies as features, in order to achieve a respectable classification accuracy. Furthermore, it is important to find out the importance of each feature in order to classify an email as a spam/ham. This leads us to use Relief Algorithm in order to retrieve the optimum weights of each feature which will be used to formulate the classification model produced by the Naïve Bayes Classifier.

Keywords: spam, ham, Naïve Bayes classification, n-gram model

INTRODUCTION

In today's gullible world, the spammers are generating immense amount of profit while hurting the email receivers and the businesses providing the medium to send emails. This phenomenon has been exploited resulting in extreme losses to both the victims and is not showing any signs to stop but is increasing as time passes [1].

LITERATURE

Naïve Bayes Classifier

In order to build an automatic classification system, we need a method that is both effective and less resource consuming for quick classification. Popular opinion suggests that Naïve Bayes Classification just fits the requirement since the training procedure takes linear time. Naïve Bayes Classifiers are built on one common assumption that all features are independent of each other, that is they are conditionally independent of each other given the class.

Given a vector of independent features $x = (x_1,\ldots,x_n)$, the classifier assigns a sample probabilities $p(C_k|x_1,\ldots\ldots,x_n)$ or $p(C_k|x)$ for each of k possible classes C_k. Further, using Bayes' theorem, the above probability can be further represented as

$$\frac{p(C_k)p(x|C_k)}{p(x)}$$

Using the joint probability model and the fact that all the features are conditionally independent [2],

$$p(C_k|x_1,\ldots\ldots,x_n) = p(C_k) \prod p(x_i|C_k)/p(x)$$

The above probability model is converted into a classifier by assigning the sample with the class of which the probability is maximum.

Since we are using bag of words model, where the frequency of n-gram words build up the feature, we are using a Multinomial Naïve Bayes approach in order to generate a classifier model for detecting spam emails.

Zero Probability Smoothing

The most primitive way of calculating $p(x_n|C_k)$ is to find out the percentage of the samples with the given class and the given feature(n-gram):

$$p(x_n \mid C_k) = count(x_n, C_k)/count(class)$$

But this approach has its own drawback where a feature never occurs with the given class. For example, when a 2-gram feature is being considered, given the class being spam, it is possible that ***count (x="chicken sale", C = "spam")*** is equal to zero. This will cause the class likelihood for spam to be zero. Hence this will result in the input never being assigned in this label irrespective of how well that feature is proving to be important in the classification process. To amend this, we are using Heldout Estimation [3] where number of 'missing' feature values for each (x, class) pair is retrieved and is added in the count of the unseen event.

Relief Algorithm

After creating the classification model of spam filtering, the next important step is to choose features which are conducive to the Bayesian model which generate feature values. Hence a feature subset selection seems necessary in order to form a decision about which features to retain, which to discard and how much influence should be assigned to each retained feature. For this task of estimating the quality of features and their relative importance when used together in classification, we implemented the primitive version of Relief Algorithm which works for classification problems with two classes. Since the current problem description involves only two classes and a vector of features, we selected the most primitive version of Relief Algorithm, although the algorithm has evolved and gotten sophisticated in handling more than two classes. The algorithm searches the two nearest neighbours, one from the same class (nearest Hit) and the other from a different class (nearest Miss). The nearer the feature value of an instance to the nearest Hit and the farther the feature value of that instance to the nearest Miss, the better the quality of the feature. Following is the pseudocode of the algorithm that we implemented [4]:

Input: for each training instance a vector of attribute values and the class values
Output: the vector W of estimations of the qualities of attributes

1. set all weights W[A] := 0.0;
2. for i:=1 to m do begins
3. randomly select an instance R_i;
4. find the nearest hit H and nearest miss M;
5. for A:=1 to a do
6. $W[A]:=W[A] – diff(A,R_i,H)/m + diff(A,R_i,M)/m$;
7. End

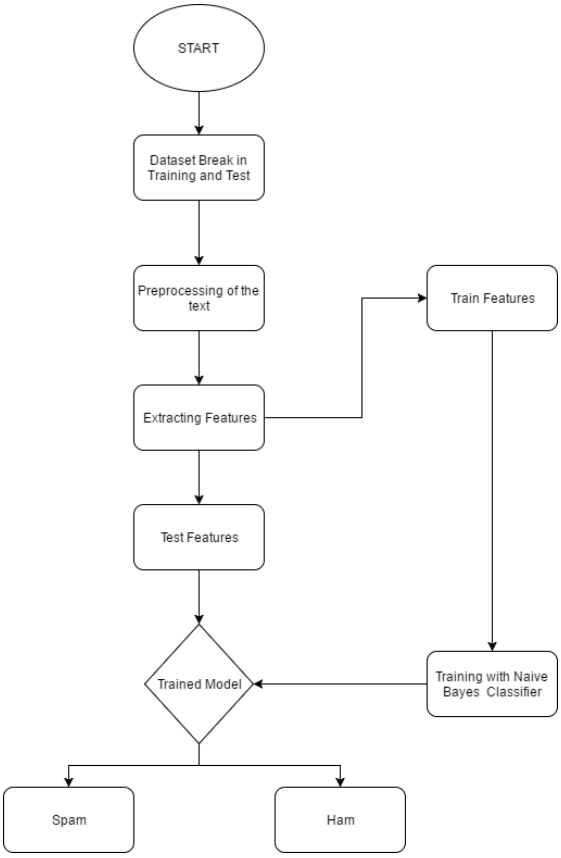
F-Score

We have used F-Score in order to diagnose the accuracy of the classifier where it is given by:-
F-Score = 2pr/(p+r)

Here 'p' stands for precision and 'r' stands for recall. Precision is the count of correctly classified samples divided by the total number of classified samples. Recall is the count of correctly classified samples divided by the total number of samples.

PROCEDURE

In this section, we illustrate our method for classifying a corpus of emails provided by Enron Corpus [5]. We are running the procedure over each 6 datasets separately provided by Enron.



Dataset breakup

Given a dataset, we are dividing it into a training set and testing set in a 30:70 ratio since we don't want to bias the classification results and results should be generalizable. This results in four sets of data – Spam-Training (30% of spam emails), Spam-Testing (70% of spam emails), Ham-Training (30% of ham emails) and Ham-Testing (70% of ham emails). All of these sets are sent to a pre-processing module with the aim of improving the feature quality used for classification.

Pre-processing

Each element of an email can be categorized as a number, attachment, all-capital words, html links and words. For each list of files from the above dataset breakup, we are going through a routine where each file is tokenized into the above mentioned categories. Every number, attachment and html link, regardless of what they are constituted of, are given the same label corresponding to their category and added to the list of tokens. In the case of an all-capital-word, its category label is added to the token list and a lower-

cased version of the word (after lemmatizing it) is also added to the token list. The remaining category of elements are lemmatized and added to the list of tokens unless the elements are punctuations or fall into the category of stop words (for example: “a”, “the” etcetera).

Feature Extraction

After generating a token list, given an ‘n’, we are creating an n-gram model for the email, where we are combining every adjacent n elements. For example in a 2-gram model, given a spam email and its token list = [“goal”, “football”, “add”], the following dictionary is created – [[[“goal”, “football”], true], [[“football”, add”], true]]. With every dictionary created, a corresponding list of tuples is created with every element mapped to the file’s respective pre-classified class. For each dataset, each file’s list of tuples are combined together forming one global list of tuples which will be used in generating conditional probabilities as described in the Naïve Bayes Classifier section of Literature.

Training

Using the two dictionaries generated above (Spam-Training, Ham-Training), we are training the model in order to create a list of probabilities with each element being $p(x_i \mid C_k)$ where x_i is the n-gram token and C_k being spam/ham. In the cases of unseen event, their probabilities are generated using Zero Probability Smoothing as describes previously.

Classifying

Given the two datasets (Spam-Testing, Ham Testing) and the above generated Naïve Bayes model, we proceed towards generating spam and ham probabilities for each file using the following equation:-

$$p(C_k|x_1,\ldots\ldots,x_n) = p(C_k) \prod p(x_i \mid C_k)/p(x)$$

as described in the Literature section. All emails are classified by taking the class of which it has a larger probability than the other.

FEATURE WEIGHING

After running the above procedure, we proceed to the feature weighing routine. Here we want to evaluate each n-gram feature (1<= n <= 3) and generate a model using the quantified evaluation with the hope that this model would outperform all the single-featured probability models generated previously.

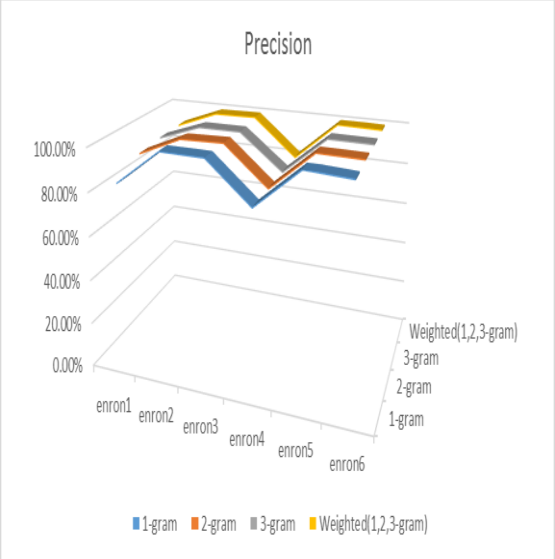
We are passing 1-gram, 2- gram and 3-gram feature distribution of both the classes (spam, ham) to the Relief Algorithm designed as per the algorithm described above. Further, we generate weighted values of each feature which is later used in generating a classification model based on weighted averages of the features.

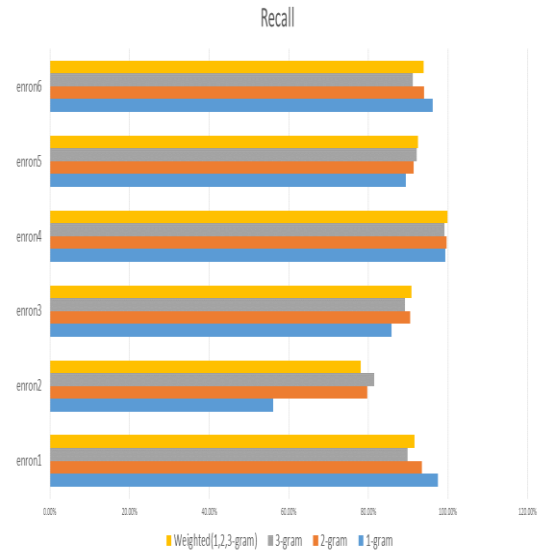
RESULTS

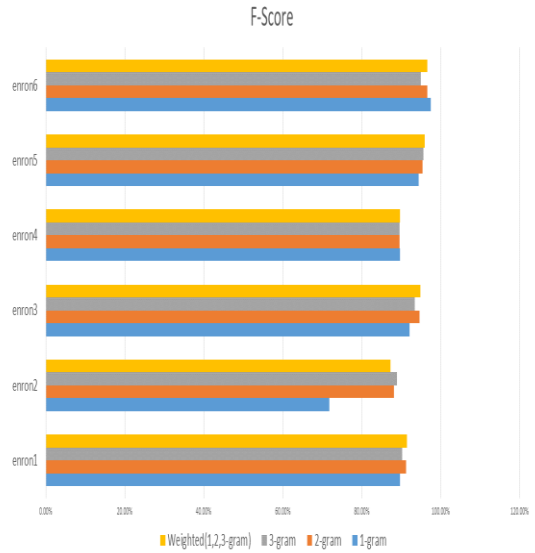
The following graphs compare accuracies of each classification models:-

1. 1-gram
2. 2-gram
3. 3-gram
4. Weighted average of n-gram (1<=n<=3)

for all the 6 email corpuses provided by Enron.







Dataset	1-gram	2-gram	3-gram
Enron1	0.043933999826441784	0.1800385042935387	0.3621209569985801
Enron2	0.4822925929369956	0.5041072266852856	0.4965818528789872
Enron3	0.4980216053587921	0.8592243668725497	0.7375805609365669
Enron4	0.6804112230033307	0.7702767951867431	0.7686327527742408
Enron5	0.4052633505860674	0.4755698882082974	0.4864704678464767
Enron6	0.06258922687538522	0.6431867554908086	0.46141849256261874

CONCLUSION

From the above F-Score graph, we can notice that Enron6, Enron5, Enron4 and Enron3 datasets are having higher weighted average model score as compared to single-featured models.

Further, the above table suggests 2-gram and 3-gram models have higher weights than 1-gram model. The F-Scores can be increased further by adding more emails into training set. Hence, Relief Algorithm is generating meaningful weights of each feature which is ultimately raising the F-Scores resulting in greater accuracy of the spam filter.

REFERENCES

[1] https://www.sophos.com/en-us/press-office/press-releases/2008/07/dirtydozjul08.aspx
[2] https://en.wikipedia.org/wiki/Naive_Bayes_classifier#Introduction
[3] http://www.nltk.org/book/ch06.html
[4] http://lkm.fri.uni-lj.si/rmarko/papers/robnik03-mlj.pdf
[5] http://www2.aueb.gr/users/ion/data/enron-spam/
[6] Androutsopoulos, J. Koutsias, K.V. Chandrinos, George Paliouras, and C.D. Spyropoulos (2000). An Evaluation of Naive Bayesian Anti-Spam Filtering.
[7] Bay, S.D. (1998). Combining nearest neighbor classifiers through multiple feature subsets. Proceedings of the 7th International Conference on Machine Learning (pp. 37-45), Morgan Kaufmann.
[8] Kononenko, I. (1994). Estimating attributes: analysis and extensions of Relief. In De Raedt, L. and Bergadano, F. (Eds.), Machine Learning: ECML-94 (pp. 171-182), Springer Verlag