# Project Report - CS 6364 (Artificial Intelligence)

Submitted by: Anupam Gangotia

Project Title: Email Classifier

Abstract:

This article gives a brief overview of how we can use a hybrid of decision tree and naive bayes in classification of emails. A brief description of the activities involved for text classification is presented. Today everyone receives hundreds of mail every day, and organizing these emails becomes a chaos for the receiver. This Problem becomes even worse with the presences of Spam Mails. The proposed solution is built upon datasets from CSDMC2010 SPAM corpus and 20 newsgroups data-set. The article is related to the author’s attempt of applying the nbtree in text classification, and may therefore be of interest primarily to those getting acquainted with text classification using machine learning.

Introduction:

Here are some stats about emails received by us every day.



Reference : Email Statistics Report, 2011-2015 (Editor: Sara Radicati, PhD; Principal Analyst: Quoc Hoang)

Electronic spamming is a major problem these days for email users. Spam email is an email sent to somebody without consent and its content can cause unease and distress. Few characteristics for spam emails are:

• The spam email is unsolicited.

• Always sent in bulk and the sender of spam email doesn’t target recipients personally.

Thus, the addresses of recipients often are guessed and the same spam email is sent to numerous people at the same time. These spam emails have already caused many problems such as consuming network bandwidth, wasting recipient time and so on. To resolve these problems, classiﬁcation of spam email from legitimate email has become very important. Recently, many machine learning and data mining techniques have been applied in spam email classiﬁcation, such as Naive Bayes, SVM or Decision Tree.

**Prior Art:**

Lots of research has already been done on using Machine Learning Techniques for email classifications. Paper by Konstantin Tretyakov on “Machine Learning Techniques in Spam Filtering” has provided a nice overview of some of the most popular machine learning methods (Bayesian classiﬁcation, k-NN, ANNs, SVMs) and of their applicability to the problem of spam-ﬁltering.



Performance measures from the “Machine Learning Techniques in Spam Filtering” by Konstantin Tretyakov

So clearly this paper suggests Naive Bayes and SVM are a good choice in spam filtering.

Further Paper by Muthukaruppan Annamalai, Ankur Jain, Vaishnavi Sannidhanam on “A Novel Hybrid Approach to Machine Learning” shows how we can combine decision trees along with naive bayes to create a more effective solution towards spam filtering. This paper clearly stated how decision tree performs better than the naïve bayes on small dataset, and how naïve bayes outperforms decision trees in case of large datasets. Thus making hybrid approach a middle path between the two choices.

To classify the emails, we have used a hybrid approach that combines decision tree and naive bayes classifiers together. The naive bayes classifier performs reasonably well. Its biggest weakness lies in the assumption that the attributes are independent of each other. Decision trees classifiers, unless pruned, even for a small number of attributes require prohibitively large memory and running times.

Therefore our approach combines the Decision Tree Classifiers' propensity to separate out dependent attributes, and the effective classification by the Naïve Bayes Classifier on independent attributes. The idea is simple. In the learning phase, the Hybrid Classifier grows a tree exactly like the Decision Tree. The only difference is at the leaves, where a naive Bayes learner is now implemented. This Naive Bayes learner learns only on the training examples that arrive at that particular leaf, using only those attributes that have not been used by the Decision Tree along the path from the root to the leaf. During the inference phase, just as in Decision Trees, the attribute values of the test example determine the path that it takes down the tree and hence the particular leaf node that it reaches. The decision at the leaf node is taken by the Naive Bayes classifier based on the attributes of the test which have still not been considered.

**Proposed Solution:**

1. Architecture: Architecture of the system is divided into three main phases namely initialization, classification and user feedback.



* 1. Initialization: we initialize separate instances of NbTree hybrid class, for each of the target category(spam, atheism, medical, autos, sports). During Initialization, there are two major tasks text pre-processing and Feature Vectors creation. These feature vectors are then used by NBTree Algorithm as a training data set.

Text Pre-processing:

- Tokenization using Stanford Parser.(Also omits unnecessary parts of speech.)

- Stop word removal. : To remove words such as “a”, “the”, “I”, “he”, “she”, “is”, “are”, etc

- Normalize words. : Stanford Lemma

- Then make instance based on top K term frequency.

Feature Vectors creation: converting all messages to vectors of numbers called feature vectors.



* 1. Classification: we read the test instances, and try to classify it. For classification we first classify it as Spam / Ham. If the result is specified as Spam, we stop here and return this as class of email. If not then we classify the test amongst the remaining classes and return the class which has the highest probability.

If none of the classifiers returned true, then we classify this message as Ham.



* 1. User Feedback: we simply accept the user feedback about the result. Then update our knowledge bank with user feedback.



1. Implementation Details: About implementation there are two major things to talk about, NBTree algorithm and naive bayes text classification using multinomial NB model.
	1. NBTree algorithm:
		* 1. For each attribute Xi , evaluate the utility, u(Xi ), of a split on attribute Xi .
			2. Let j = argmax (ui ), i.e., the attribute with the highest utility.
			3. If uj is not significantly better than the utility of the current node, create a Naive-Bayes classifier for the current node and return.
			4. Partition the set of instances T according to the test on Xj . IfXj is continuous, a threshold split is used; if Xj is discrete, a multi-way split is made for all possible values.
			5. For each child, call the algorithm recursively on the portion of T that matches the test leading to the child.

Ref : “Scaling Up the Accuracy of Naive Bayes Classifiers,a Decision Tree Hybrid” by Ron Kohavi

* 1. Naive Bayes text classification: According to multinomial Naive Bayes, the probability of a document 'd' being in class 'c' is computed as



where P(tk|c) is the conditional probability of term tk occurring in a document of class c.

P(c) is the prior probability of a document occurring in class c.

In text classification, our goal is to find the best class for the document. The best class in NB classification is given by cmap



The above equation can result in a floating point underflow as many conditional probabilities are multiplied. Hence we take log to avoid such situations.



Also P(tk|c) is calculated as (Count(tk in Class c)+ 1)/(Count (c) + |V| ).

Where Count(c ) =Total words in Class c, V=Dictionary size

Here we have use Add-1 smoothing, just to avoid the case when a particular word is not present in our knowledge bank.

1. Problems Faced: Earlier I was using Porter stemming algorithm for removing the commoner morphological and in flexional endings from words. However results were not that good with Porter Stemming algorithm. Hence decided to go with the lemmas provided with Stanford Core NLP.

Finding Training data, was also quite difficult. For Spam classification numerous Spam corpus are available, but for other categories there are very few. Finally found 20 news-group dataset from (<http://www.csmining.org/index.php/id-20-newsgroups.html>) but this data was not in the form of emails. So text pre-processing was needed before this can be used in our system.

Some python scripts were also used, to prepare different sets of data sets for testing.

Results:

* + 1. Observation 1: Hybrid Classifier used as Binary Classifier against test data same as training.

|  |  |  |
| --- | --- | --- |
| Training instances : 100 of each class |  | Test Instances: Same as training |
|  | Goal Class | TRUE | FALSE | Unable to identify Or Unable to read | Total Test cases | False Positive |
|  |  |  |  |  |  |  |
|  | Spam | 84 | 10 | 6 | 100 | 4 |
|  | Atheism | 98 | 2 | 0 | 100 | 1 |
|  | Autos | 96 | 4 | 0 | 100 | 1 |
|  | Medical | 99 | 1 | 0 | 100 | 0 |
|  | Sports | 98 | 2 | 0 | 100 | 1 |



Test Instances classified when test data is same as training data

* + 1. Observation 2: Hybrid Classifier used as Binary Classifier against test data different from training.

|  |  |  |
| --- | --- | --- |
| Training instances : 100 of each class |  | Test Instances: Different from Training Instances |
|  | Goal Class | TRUE | FALSE | Unable to identify Or Unable to read | Total Test cases | False Positive |
|  |  |  |  |  |  |  |
|  | Spam | 46 | 46 | 8 | 100 | 40 |
|  | Atheism | 17 | 83 | 0 | 100 | 50 |
|  | Autos | 33 | 67 | 0 | 100 | 50 |
|  | Medical | 38 | 62 | 0 | 100 | 49 |
|  | Sports | 25 | 75 | 0 | 100 | 53 |



Test Instances classified when test data is different from training data

* + 1. Observation 3: Hybrid Classifier used as Multi Class Classifier against test data same as training.

|  |  |  |
| --- | --- | --- |
| Training instances : 100 of each class |  | Test Instances: Same as Training Instances |
|  | Goal Class | Spam | Ham | Atheism | Auto | Medical | Sports | Unable to identify Or Unable to read | Total Test cases |
|  |  |  |  |  |  |  |  |  |  |
|  | Spam | 84 | 7 | 0 | 1 | 2 | 0 | 6 | 100 |
|  | Atheism | 45 | 0 | 15 | 10 | 10 | 20 | 0 | 100 |
|  | Autos | 41 | 2 | 7 | 19 | 15 | 16 | 0 | 100 |
|  | Medical | 39 | 1 | 10 | 14 | 16 | 20 | 0 | 100 |
|  | Sports | 29 | 0 | 16 | 15 | 23 | 7 | 10 | 100 |

* + 1. Observation 4: Hybrid Classifier used as Multi Class Classifier against test data different from training.

|  |  |  |
| --- | --- | --- |
| Training instances : 100 of each class |  | Test Instances: Different from Training Instances |
|  | Goal Class | Spam | Ham | Atheism | Auto | Medical | Sports | Unable to identify Or Unable to read | Total Test cases |
|  |  |  |  |  |  |  |  |  |  |
|  | Spam | 51 | 9 | 6 | 10 | 7 | 10 | 7 | 100 |
|  | Atheism | 43 | 9 | 2 | 18 | 16 | 12 | 0 | 100 |
|  | Autos | 38 | 9 | 12 | 8 | 17 | 15 | 1 | 100 |
|  | Medical | 49 | 14 | 9 | 5 | 5 | 18 | 0 | 100 |
|  | Sports | 28 | 12 | 14 | 21 | 20 | 4 | 1 | 100 |

 In observation 1, results are satisfactory. In observation 2, results can be improved if we improve our data set, and use more training instances. However sufficient improvements are needed in case of observation 3 and 4. Currently this algorithm doesn’t take account of pruning or cross-validation which might be the cause of unsatisfactory results in observation 3 and 4 along with low number of training sets.

Conclusion:

The aim of this project is to deliver software which is able to classify the emails into different categories. In this document a comprehensive approach towards developing a text classification engine is described. This approach has used a hybrid approach towards text classification, which is a mix of decision tree classifier and naive bayes classifier. We start by calculating utility of a split on an attribute, and select the attribute with highest utility value for the split. After partitioning we keep on doing the same thing recursively, till we reach a point where split does not increase our utility. To calculate the utility, we have used naive bayes probability value. Since probabilities are multiplied in naive bayes which result in floating point underflow, so we take log of probabilities values. To deal with unknown tokens we have used “Add 1 smoothing”.

This hybrid approach is suitable when many attributes are relevant for a classification task, yet the attributes are not necessarily conditionally independent. This approach is however quite slow as its running time is longer than the decision tree algorithm or naive bayes algorithm.

* 1. Pending Issues: Some of the pending issues are
		+ - Speed Improvement issues.
			- Features selected after text pre-processing still consist of junk words.
			- For Binary classifications results are good, but for multi class classifications, still improvement is needed.
	2. Future Improvements : Some of the future improvements are:
		1. Use of n gram for features selection.
		2. Use of sender Block list, which immediately classifies the message as spam if it’s from a known spammer.

Acknowledgements: I want to thank Dr.Mithun Balakrishna for his superb guidance and valuable suggestions throughout the semester. I thank Ron Kohavi, Konstantin Tretyakov, Muthukaruppan Annamalai, Ankur Jain, Vaishnavi Sannidhanam for their work in the machine learning. I am also thankful to our Java library providers:

* 1. Stanford Natural Language Processing Group.
	2. jsoup: Java HTML Parser.
	3. Apache Log4j.

Refrences:

1. Scaling Up the Accuracy of Naive Bayes Classifiers,a DecisionTree Hybrid by Ron Kohavi.

2. Machine Learning Techniques in Spam Filtering by Konstantin Tretyakov.

3. An Efficient Two-phase Spam Filtering Method Based on E-mails Categorization by Jyh-Jian Sheu.

4.A Novel Hybrid Approach to Machine Learning by Muthukaruppan Annamalai, Ankur Jain, Vaishnavi Sannidhanam