Study and Implementation of an SMS Spam classifier

Dilip Kunderu
UFID:19601982
Department of Computer & Information Sciences and Engineering
University of Florida

CONTENTS

***Abstract*— An SMS(Simple/Short Messaging Service) Spam classifier is usually employed to filter out unsolicited text messages. At the very basic level, it can function as a binary classifier, working at the application level, in parallel with the SMS application of the users' devices. Primarily applying the naive Bayes Classification algorithm, the classifier would be improved by the Support Vector Machine mnemonics to drive the accuracy up. A spam dataset from the UCI Machine Learning Repository. Preprocessing and feature extraction are run over the DB, the techniques are independently run and the one whose results have a higher accuracy, would be taken up for further development.**

I. INTRODUCTION

Though much of the digital spam encountered all over the world is via the electronic mail, SMS as a reach-out platform is fast catching up. Since around 2013, there have been an explosive growth rate in the usage of this platform. Various web services too, via the instant messaging paradigm or related, have a service offering mimicking the one that of the SMS.

Main forms of spam is the advertising that commercial entities send to prospective customers. Secondly, nefarious activities with potentially criminal intent are also on the rise. Smartphone ubiquity has had an unwanted side effect of defrauding people by hacking their devices, whose starting points are almost always spam messages. On a subtler note, atleast here in North America, having multiple email IDs for different spheres of life maybe feasible, but having different cell phone numbers in that way is not, at least economically. Thus, although they encounter more spam(volume-wise) in email, most of the targeted spam is received via a mobile connection, as many users are tethered to one access point ie., their phone number.

While it is a significant problem, not many solutions currently exist to address this. Hence, there is a need for the effort of this project. Additionally, the datasets available pertaining to spam SMSes are very limited. Further, the language used in SMS tends to be very informal, with lots of short-handed instances of words, which may/ may not belong to a well-defined dictionary. Thus, if techniques similar to the ones used in email spam filters, their performance would be seriously degraded.

A sample database of 5774 SMSes from a dataset available at the UCI Machine Learning Repository, is used, which is a heterogeneous mixture of spam and non-spam(ham) messages. The naive Bayes approach and the SVM approach are applied after pre processing and feature extraction.

II. DATASET

The SMS Spam collection version-1 is a public dataset available from the UCI Machine Learning repository, with 6000 real messages, categorized as ham or spam. It is primarily in English. It consists of contributions from three broad sources :

- A public forum encouraging SMS users to declare their conclusion about specific text strings they have received
- Data volunteered from the students of a specific university in course of a study conducted. The primary medium of instruction for these students may/ may not have been English.
  - This point specifically highlights the fact that the number of parameters required for proper training of the system are manifold; one or two parameters are hardly sufficient.
  - E-mail based communications do have a highly diverse agendas, but to a significant extent, are formal in nature. SMS, by contrast, has less premise to be that formal.
  - The English alphabet is very commonly used to communicate in native languages, and this practice is extremely prevalent, which further garbles the accuracy.

- A corpus created of over 1000 messages collected over a period of time by an individual.

The diversity is got in by supposedly including free data collected from real users in 2014, followed by actual instances of e-marketing and advertising.

The variance in the dataset is as follows in Table :

Spam	Ham	Total
747	4827	5574

While the dataset is a RAW data, our preprocessor step will configure it into a CSV or a TSV file, with the label and the corresponding raw message divided into two columns.

The primary step in developing the dataset is the preprocessing stage wherein the training part of the dataset is tagged *ham* or *spam* for each of the text strings.

Table illustrates the training data fed into the system. This clearly shows the tagging for univariate categorization of the data

Tag	String
spam	dial 455660005 to win $34million
ham	The ice cream at Cream Stone!! It's like, BONKERS!!

TABLE I

TABLE TO TEST CAPTIONS AND LABELS

While the testing data looks like Table that follows :

String
Nhi bhai kal hi ka date hai. Koi nhi ho jayega. Aa ja.
Pls check fr 2 things before buying any house in future:1) Chain of the Property should exist.2) Mapping of the Property should be minimum <#> yards. This is theeligibility of the Property to be Valid for Mapping.

TABLE II

TABLE SHOWING TEST DATA

As we can see, the differentiation between *ham* and *spam* is not that apparent. In the present times, targeted advertising has grown much more sophisticated that basic classifiers like MNB and SVM cannot predict them properly. Multivariate versions of the classifiers would only do a slightly better job because most of the data's context is not known to the system ie., the same string could be useful for a particular person, but may seem garbage4 to someone else.

III. OVERVIEW

The stages of the project are as follows :

A. Preprocessing

- The dataset combination employed here is broadly categorized roughly into 67% training corpus and 33% testing corpus.
- The (eventual)CSV file input can be considered to be tokenized properly. Also, many of the special characters in the string (during this phase) are disregarded.
- Elements of any alphabet, other than the English alphabet, is considered a special character, and hence, is overlooked.
- Frequency of words is also stored as a metric.
- As the token count is mentioned, the tokens whose frequency falls outside the ambit of 5–500 are dropped from consideration
- The Corpus would be processed into a *Comma Separated Value*(CSV) or a *Tab separated Value*(TSV), with the training dataset including the label for the raw message.

B. Feature Extraction

As with every Machine learning algorithm, and especially the ones dealing with pattern recognition, the evolution of the feature space is of prime importance.

- Cases where the frequency is above 500 could usually be noise
- Cases where the frequency is below 5 could be considered outliers, thus minimizing the deviation from a favorable normalized state of the dataset

The frequency parameter is the base case of feature vector decisions. Multiple vectors can be generated using mappings of more specific freqencies, like proper noun counts, capital word to formal word mapping *et cetera*.

On the basis of this feature, the three specific vector parameters utilized for the feature extraction are :

- the $ symbol count
- number of digit-based literals
- String length

These three new features are also considered along with the other 1500 odd potential features. The *k way* algorithm is then applied for feature extraction.

The string length parameter is of some significance as bots that synthesize spam have a pseudo-randomly determined string length for optimal data packaging and transmission.

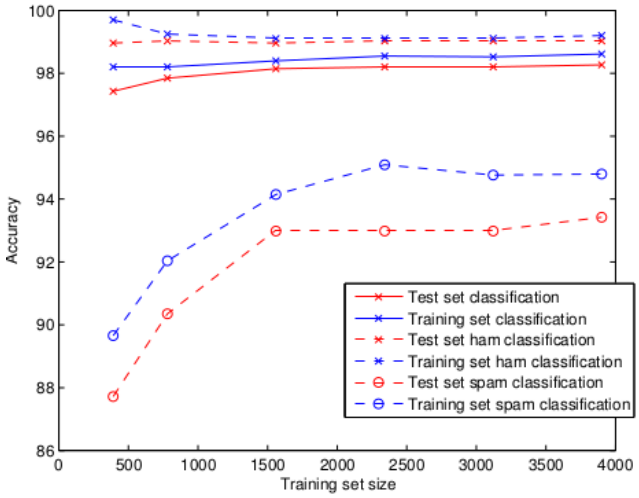
The feature extraction is roughly done on the lines of the following :

After the feature restriction and the preprocessing steps are done, up to 1551 unique features can potentially be extracted.

The following graph illustrates the accuracy of the naive Bayes classifier over different categorizations :

Algorithm 1 k-feature extraction

procedure FEATURES(DB, E, I)
    $ExtractVocabularyEV \leftarrow DB$
    $makeemptylistLst$
for each of v *in* EV :
    do $A(v,e) \leftarrow compute\_fv(DB,v,e)$
    $Lst.append(EV, < A(v,e), v >)$
    return $LargestValueFeatures(Lst,i)$

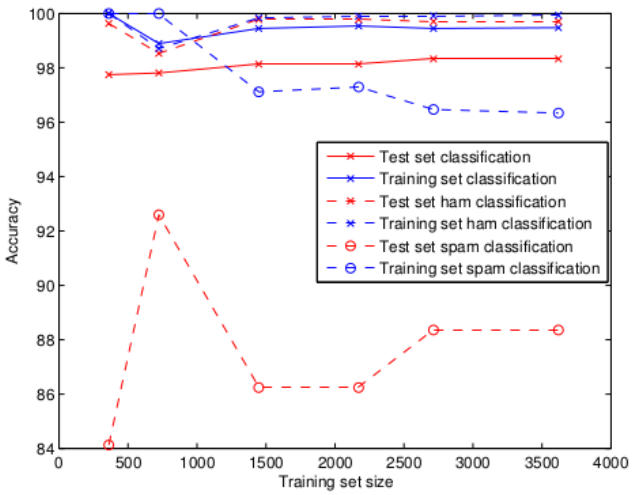


This graph was one of the main pre-cursors for the decision behind choosing the character frequency delimiters for feature extraction.

A similar analysis was done using the support vector machine algorithm, with a single class. Theoretically, the SVM would be a more ideal algorithm for the analysis of the SMS dataset as

- The corpus(under consideration here) is not that big of a dataset; it in fact can be considered minute, which suits SVM as the splicing works best on smaller datasets
- Support vectors are generally the co-ordinates of the data entity in a feature space, and so SVM's primary task, which it does well enough, is segregation. As the parity (*class*) of our analysis is just *1* ie., it is a univariate classification, its usage can be considered more appropriate.
- Since our preprocessing was not very specific, SVM initially would seem a better alternative as the number of potential features is very high. But these potential features aren't exactly the kind of ones which can classify the data(linear separation in the feature space and/or the standard deviation in the distances, is not really appreciable over most of these features).
- Further, splicing as a technique for univariate classification needlessly takes up lots of resources and thus slows the system. Also, as detailed in the introduction, the amount of noise present in datasets like the SMS dataset is very high, which puts the SVM algorithm off; noise is one of the main causes of erroneous output for SVM.

The following plot illustrates the slightly reduced accuracy for the same subset of the dataset on which the accuracy test run was done for the NB classifier :



C. Vectoring

While word-to-vector technique is not really applicable in a supervised learning context, vectoring is an important step for feeding the data as most of the algorithms take in numerical equivalents for any and every feature.

Feature vector matrices are generated over each of the three parameters. To fine tune and fast-track the training phase of the system, an ensemble technique of deploying a random forest classifier for realizing feature preferences via a weighted iteration over the feature vectors, as the possible feature set is large. But the accuracy for such a small dataset is generally not that significantly changed with this.

D. Multinomial Naive Bayes

The multinomial version of the naive Bayes classifier is employed as the occurrence frequency is the basis for the feature extraction.

The basis of the naive Bayes classifier is illustrated by the equation :

$$P(M/C) = \frac{P(C/M) * P(M)}{(P(C/M) * P(M)) + (P(C/H) * P(H)} \quad (1)$$

M : message is spam
C :word C is in the message
H : message is not spam

with the basic philosophy of the weighted feedback from the following adjustment.

$$post = \frac{priori * likelihood}{occurance} \quad (2)$$

Since the data is single dimensional, the *Max A Posteriori* rule to assign the corresponding label to the raw message.

$$C^{\text{bayes}}(x) = maxargP(X = x|Y = (1|0)) \quad (3)$$

This classifier feedback improves the training by adding a useful posterior metric.

The training of the system is done as follows :

Algorithm 2 TRAINING

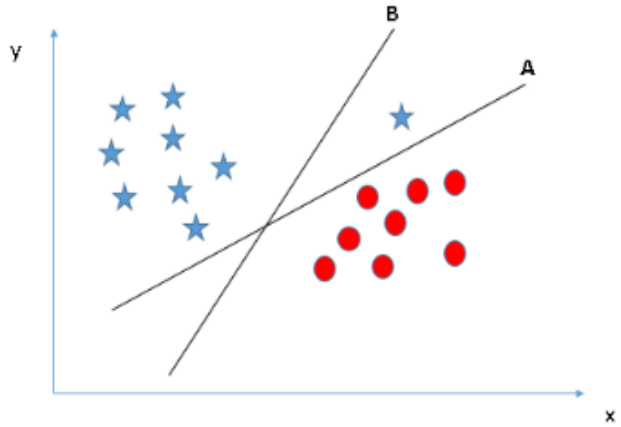
procedure TRAINSYSTEM(X, Y)
  $Vec \leftarrow extract_features(Y)$
  $Num \leftarrow frequency_return(Y)$
  for each of d *in* D :
  do $Num_d \leftarrow CurrCount(Y, d)$
  $list\_pr[d] \leftarrow Num_d/Num$
  $str_d \leftarrow AllInClassConcat(Y, d)$
  for each of u *in* Vec :
  do $U_du \leftarrow TermToken(str_d, u)$
  for each of u *in* Vec :
  do $bayesProbability[u][d] \leftarrow \frac{U_du + 1}{\Sigma_{u'}(T_{du'} + 1)}$
  return $Vec, list_pr, bayesProbability$

procedure MNB(X, VEC, LIST_PR, BAYESPROBABILITY, Y)
  $Word \leftarrow TokenExtract(Vec, y)$
for each of d *in* D :
do $wscore[d] \leftarrow addToLoglist\_pr[d]$
  do $wscore[d]$ += $addToLogbayesProbability[u][d]$
  return $argmax_{d \in X} wscore[d]$

E. Support Vector Machine

As highlighted earlier, more often than not, this classification algorithm is unsuited for univariate classification of datasets with significant amount of noise. That said, there are so many optimized implementations of the algorithm that testing them on a dataset as simple as this would not make sense; the dataset should have the potential do bolster multivariate mosaic of data clustering, segregation aside. In the current effort, we use a very simplistic SVM over a single class.

Every dataset element in an SVM environment maps to a unique point in the n-dimensional feature space, with *n* being the number of features extracted. The aim is to arrive at a hyper plane that best differentiates the point clusters.



In mathematical terms,

$$|\beta_0 + \beta^\tau \chi = 1|$$

where
$\beta \equiv weightmetric$
$\beta_0 \equiv bias$

represents a general equation of a hyperplane. The ideal hyperplane is the one whose eigen distance from clusters is the maximum ie., it

- properly separates clusters
- is as distant as possible from *all* of the clusters

That is,

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}||\beta||^2 \Big|_{\mathbf{y_i}(\beta^\tau \chi_\mathbf{i}) \geq \mathbf{1} \forall \mathbf{i}}$$

where

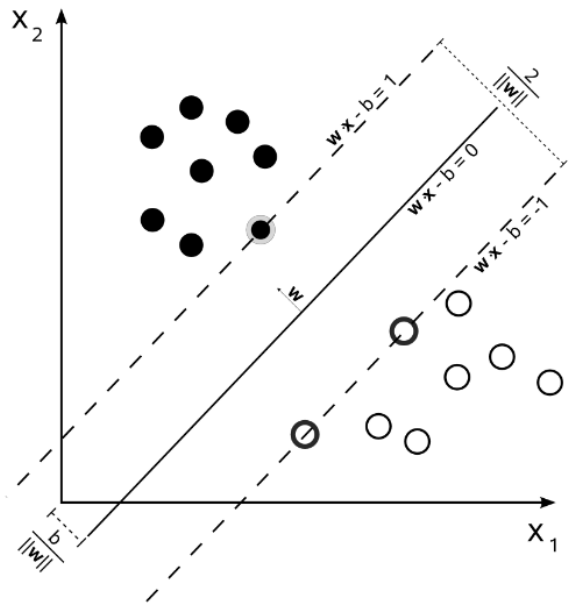
$$y_i \equiv traininglabels$$

An overview of the SVM analysis is as follows :

Algorithm 3 SVMEnsemble

procedure SIMPLE_SVM()
  $Candidates \leftarrow \{pairswithlargesteigendistance\}$
  while violation of class separation exists **do**
    Get reference to violator
    add violators to the candidates set
    if weight param ¡ 0 **then**
      drop p from candidate

An illustration below shows us the decision-making process for appropriate support vector candidates is given below :



The dotted lines represent the support vectors in perspective, which are synthesized on the basis of the hyperplane (the solid line) amongst the two classes (univariate differentiation)

F. Ensemble method Appending

If time permits, amalgamation of an ensemble of Random forest classifier and/or Adaboost to the SVM-infused model can be done. Most importantly, the model would be trained against unlabeled data and checked for accuracy, and parameters in the SVM tweaked for improving accuracy. For example, any word/phrase written in English but might actually belongs to the vernacular, it usually gets categorized as spam. Trying to train the model in such a way that it can get at least 50% of such occurrences get categorized as *ham* would be the aim. Maybe the model could be tweaked to accomodate the feature of training itself from unsupervised data too.

One other aspect for which the Random forest classifier might be useful if

- label(s) pertaining to the sender are also included in a given dataset, for a more efficient categorization
- unsupervised learning phase might involve manual intervention initially to correct classes of categorized messages

Illustrating the second point : a message usually categorized as *spam* might have come from a legitimate sender or may have been an opt-in by the user.
Such nuances call for employment of multiple decision trees.

At this point, the dataset used cannot be called exhaustive, as the diversity in the lingo of text messaging matches the diversity of the user demographic, if not more. Thus, a need for the Adaptive Boosting Technique(*Adaboost*) to reinforce the weighted classifier feedback.

IV. Results

- There were two separate work flows to test the accuracy of the system. One flow is intended to run on 32% of the training data. The other flow, on the whole dataset.

- The custom flow is intended to add labels to an unlabeled corpus. Since the code is being run on a core i3 laptop, the task is too heavy owing to too many file accesses and writes.

- The test file for the regular flow is a subset of the whole training dataset. But in the case of the second flow, the test data is a bigger and randomized collection of the strings from a diverse corpus, which overlaps with the main corpus, but the subset is not necessarily overlapping.

- The regular flow goes like
  - processing the training data
  - feature extraction
  - training data vectorization
  - processing testing data
  - vectorize testing data
  - train the
    - MNB classifier
    - SVM classifier
  - test the
    - MNB classifier
    - SVM classifier

In an iterative fashion, both the classifiers are tested and the metrics recorded are

- Training duration
- Testing duration
- Accuracy

The normal flow goes like follows :

```
(venv-dir) dilip@dilip-Inspiron-3521:~/Desktop/
Pattern_Final/SMS-Spam-Detection-master$
./normal-flow.sh
Processing training data...
Now extracting features from training data...
Now building feature vectors from training data...
Now preprocessing test data...
Now building feature vectors from test data...
Now training MNB classifier...
Started MNB at:  2017-04-25 18:40:34.220850
Finished MNB at:  2017-04-25 18:40:34.354563
Time taken:  0:00:00.133713
Now testing MNB classifier...
0.9743130227
Started MNBTest at:  2017-04-25 18:40:34.543199
Finished MNBTest at:  2017-04-25 18:40:34.964958
Time taken:  0:00:00.421759
Now training SVM classifier...
Started SimSVM at:  2017-04-25 18:40:35.546455
Finished SimSVM at:  2017-04-25 18:40:47.150330
```

```
Time taken:  0:00:11.603875
Now testing SVM classifier...
0.934289127838
Started SimSVMTest at:  2017-04-25 18:40:47.345944
Finished SimSVMTest at:  2017-04-25 18:40:52.364193
Time taken:  0:00:05.018249
```

The following table illustrates the results of running the normal flow :

classifier	Train	Test	Accuracy
MNB	0:00:00.177987	0:00:00.401680	0.9743130227
SVM	0:00:11.526784	0:00:05.068104	0.934289127838

Doing the custom run did generate the following amount of tags :

ham	spam	Total
638	47341	65343

- As noted in the graphs detailed earlier, the number of false positives seems to be greater then the number of false negatives because the classifier associated many of the numerical specifics to spam, and hams with such numerical values were tagged as spam.
- There was absolutely no action taken on
  - addressing of duplicates in the corpus
  - *stemming* of any words

  Thus, bias towards spam grew higher
- The outliers belonging to languages other than English were handled by the smoothening the graph by Laplacian transforms

V. CONCLUSION

As the results table shows, there was a textbook behavior when it came to train and test the classifiers.

- The SVM classifier was much slower to train, even though the dataset is not that big
- The SVM classifier took more time for the testing phase too, owing to its *splicing* methodology for classification. Further, the presence of noise and duplicates have bungled its performance

REFERENCES

[1] http://www.dt.fee.unicamp.br/ tiago/smsspamcollection/
[2] http://cs229.stanford.edu/proj2013/ShiraniMehr-SMSSpamDetectionUsingMachineLearningApproach.pdf
[3] Wikipedia :Naive Bayes spam filtering
[4] Tiago A. Almeida, Jos Mara G. Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of SMS spam filtering: new collection and results. In Proceedings of the 11th ACM symposium on Document engineering (DocEng 11). ACM, New York, NY, USA, 259-262. DOI=10.1145/2034691.2034742 http://doi.acm.org/10.1145/2034691.2034742
[5] http://en.wikipedia.org/wiki/Mobile phone spam
[6] Wikipedia :Naive Bayes Classifier
[7] Press Release, Growth Accelerates in the Worldwide Mobile Phone and Smartphone Markets in the Second Quarter, According to IDC, http: //www.idc.com/getdoc.jsp?containerId=prUS24239313
[8] "http://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/"
[9] "http://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics"
[10] "https://www.analyticsvidhya.com/blog/2015/10/understaing-support-vector-machine-example-code/"