Email Spam Filter using Naïve Bayes classification

Engineering Project Report

Submitted in the partial fulfillment of the
requirements for the degree
of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE ENGINEERING

Submitted by

KISHORE KUMAR VADDE
R091519

Under the guidance of

VARUN KUMAR

Rajiv Gandhi University of Knowledge and Technologies

R.K.Valley, April 2015

CANDIDATE DECLARATION

I hereby declare that the work presented in this report entitled **“Email Spam Filter using Naïve Bayes Classification”**, submitted in the partial fulfillment of the degree of Bachelor of Technology (B.Tech), in Computer Science Engineering at Rajiv Gandhi University of Knowledge and Technologies, R.K.Valley, is an authenticated record of my original work under the guidance and supervision of **Lect. VARUN KUMAR**.

This project work was done in full compliance with the requirements and constraints of the prescribed curriculum.

Date: Kishore Kumar

Place: R.K.Valley R091519

CERTIFICATE FROM SUPERVISOR

I do hereby recommend that the project work prepared under my supervision by **Kishore Kumar Vadde** titled “Spam Filter using Naïve Bayes Classification", be accepted in the partial fulfillment of the requirements of the degree of Bachelor of Technology in Computer Science Engineering for Examination.

Date: Lect. Varun Kumar

Place: R.K.Valley

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor Lect. Varun Kumar for his trust in me to choose a research topic in Machine Learning and he was always there for me whenever I needed a help. And I would like to mention my friend Srinivas Reddy for stimulating me towards research areas. And finally I would like to thank my parents and my uncle who always trusted in my choices in life.

Kishore Kumar

Abstract

We are living in an era of information. Internet is playing the role of an ocean of information. We send the data across the network to other computers. Using Mails has become a daily activity in recent days. But on internet getting unwanted and malicious mails has also became common. Unwanted mails contain spam data which is unnecessary data to users. Sometimes spam data contains malicious attacks towards the system. The need of email spam filters is necessary to prevent these kinds of attacks. As a part of learning Machine Learning techniques I considered developing an email spam filter. There are various kinds of classification algorithms in machine learning but I considered working with Naïve Bayes classifiers. This family of classifiers is the part of supervised learning algorithms. In supervised learning algorithms will get trained by the data provided from previous experiments and then they will be used to predict the results for new problems which are related to them. Naïve Bayes Classifier is so simple which is based on the Bayes probability theorem but yet so much powerful to classification problems.

Contents

Abstract

1. **Introduction**
   1.1. Problem definition and Scope
   1.2. Organization of the report

2. **Machine Learning**
   2.1. Introduction to Machine Learning
   2.2. Supervised and Unsupervised Learning

3. **Naïve Bayes Classification**
   3.1. Bag of words model
   3.2. Multinomial Naïve Bayes model

4. **NLTK (Natural Language Processing Tool kit)**
   4.1. Enron email Corpus
   4.2. Stop words Corpus
   4.3. Tokenizers
   4.4. Stemmers

5. **Methodology**
   5.1. Feature Extraction
   5.2. Training Classifier
   5.3. Testing the Classifier

6. **Results and conclusion**

Chapter 1

Introduction

Spam, also known as “junk e-mail” or “unsolicited commercial e-mail”, pollutes the one’s medium of electronic mail. Left unfiltered, recipients of spam must waste time deleting annoying and possibly offensive mails. When a user is flooded with a large amount of spam, the chance of overlooking a legitimate mail increases. Spam also creates a burden on mail servers and Internet traffic, all for unwanted messages.

Researchers have examined several different approaches to spam filtering, many of which are widely used. One broad approach attacks spam at the network level. In this approach some servers block spam based on so called Real time Black Hole lists, which alert servers of undergoing spam filtering. Another broad approach examines the content of an incoming message for features which indicate it status as spam or legitimate. I am using here the latter approach towards spam content.

1.1 Problem definition and Scope

The whole email spam classification problem can be divided into three stages; they are 1.feature extraction from emails 2.Training the classifier and 3.classifying new mails according to the classification algorithm. In these phases of spam filtering feature extraction is so important to consider in the view of accuracy of classifier.

Scope of this spam filter using Naïve Bayes classification is so vast. This is so simple to implement in real time mail clients and also it gives more powerful results.

1.2 Organization of the report

The remaining study of this report organized as following manner.

- Chapter 2 gives a brief introduction to machine learning. It describes about the two different approaches of machine learning that are supervised and unsupervised learning.

- Chapter 3 explains the theory behind the Naïve Bayes Classification and also it describes the feature extracting model bag of world model. It also gives a description to calculate the feature probability using multinomial Naïve Bayes model.

- Chapter 4 gives the description about Natural Processing Tool Kit and its stop-words corpus. And also explains about the tokenizers and stemmers of NLTK.

- Chapter 5 explains the methodology used in email spam filter development. And its various stages like feature extraction, training the algorithm and finally classification.

- Chapter 6 presents the results obtained after training and testing the classifier on the publicly available email corpus and the conclusion.

.

Chapter 2

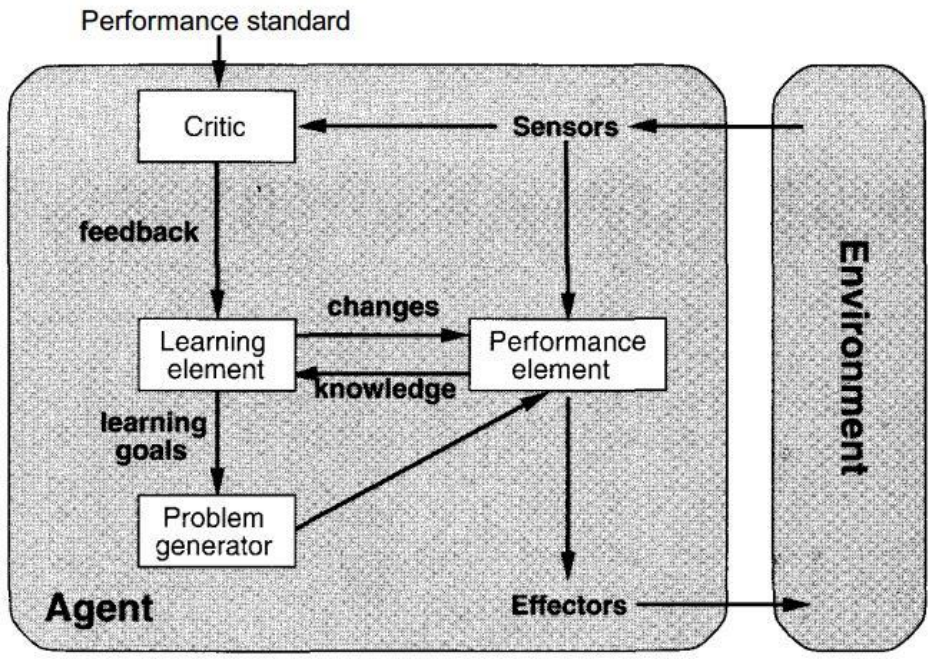
Machine Learning

2.1 Introduction to Machine Learning

Computers are modern drivers of the world. Computers are developed based on the algorithms. But mostly these algorithms are intended to do a particular task always in same manner and with same accuracy. But a branch of Computer Science which is Artificial Intelligence (AI) has changed the way the algorithms work. It deals with the algorithms which can make the system to be pretended as intelligent. AI has various sub branches in it. Among them machine learning has its own importance.

The name of the branch itself tells that making the machines to learn things. Machine learning has various approaches to feed the systems to learn various things. There is a famous definition on machine learning algorithms “A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E”. Machine learning algorithms are of different types. Among them supervised and unsupervised learning will be explained in following study.

Learning Agent



The above figure describes the learning agent program. In that **Learning element** which is responsible for improving agents performance. The **performance element** is responsible for the actions the learning agent takes on external environment. The **critic** is responsible to tell the learning element that how well the agent is performing. The **problem generator** suggests the actions that will make the agent to perform well. And finally **sensors** and **effectors** are responsible for perceiving things from and to make actions on environment respectively. Here the agent program will be given by a **performance standard** by the critic which is used to compare the agent's performance to the actual success.

2.2 Supervised and Unsupervised learning

Supervised learning algorithms are used whenever the prior knowledge of the problem is known and the output domain of the problem is also known. In Supervised learning, algorithms are given by the various inputs and their corresponding outputs to learn about the problem and its behavior. And then whenever a new problem is encountered they will estimate the outcome of it using a specific strategy derived from the prior knowledge of the problem domain. In the learning agent model the available feedback enables the agent to predict on new problems of same domain. In supervised learning Linear regression and classification are well known strategies.

Unsupervised learning algorithms are used whenever there is no knowledge regarding the problem domain and its outcomes. In Unsupervised learning, algorithms will derive a strategy based on the given problem instance. In Unsupervised learning clustering is well known problem.

Chapter 3

Naïve Bayes Classification

Naïve Bayes classifiers are the linear classifiers that are known for being simple and yet powerful. Their probabilistic model is based on the Bayes' theorem. In Naïve Bayes approach there is an assumption that the features in the data set are mutually independent. Being relatively robust, easy to implement, fast, and accurate, naive Bayes classifiers are used in many different fields. Some examples include the diagnosis of diseases and making decisions about treatment processes, the classification of RNA sequences in taxonomic studies, and spam filtering in e-mail clients.

3.1 Bag of words model

The most important thing to consider in classification problems is that extracting and selecting effective features. The three main criteria of good features is as follows

- **Salient:** The features are important and meaningful with respect to problem domain.
- **Invariant:** Invariance is described in the context of image classification; the features are unsusceptible to distortion, scaling and orientation etc.
- **Discriminatory:** The selected features have a enough information to distinguish well between the classes when used to train the classifier.

The main idea behind the bag of words model is as simple as it sounds. First consider creating unique vocabulary of features. This vocabulary contains each unique feature of training data and its count of occurrence in each document. The vocabulary can then be used to construct a d-dimensional feature vectors for individual documents, where dimensionality is equal to the number of different words in the vocabulary.

Consider an example

Document1 = ”we live In a world of internet”
Document2 = “ internet is the largest network in the world”

Vocabulary={we:1, live:2, in:2, a:1, world:2, of:1, internet:2, is:1, the:2, largest:1, network:1}

Dimension(d) = |Vocabulary|

Tokenization:

Tokenization describes a general process of breaking down the text into individual elements that serve as input to various language processing algorithms. Tokenization is accompanied with other processing steps stop words removal, stemming and lemmatization.

Consider an example:
Sentence = ”hi my name is a robot, what yours?”
Tokens = {‘hi’, ‘my’, ‘name’, ‘is’, ‘a’, ‘robot’, ‘’’, ‘what’, ‘yours’, ‘?’}

Stop words:

Stop words are the words that are particularly considered common in text corpus and thus considered as rather uninformative. To remove stop words we can find most frequently occurring words in the documents and make them as stop words. Another approach is that search against language specific stop word dictionary.

Consider example:
Sentence = “he is so smart in mathematics”
Tokens = {‘smart’, ‘mathematics’}

Stemming and Lemmatization:

Stemming is a process of transforming a word into its root form. Stemming can create non real words which are not grammatically correct. In order to avoid this problem Lemmatization is used. In lemmatization word will transform into its canonical form which is grammatically existed. But lemmatization is more difficult and computationally expensive than stemming.

Consider an example:
Sentence = “A swimmer likes swimming, thus he swims”
Stem_tokens = {‘A’, ‘swimm’, ‘like’, ‘swimm’, ‘thu’, ‘he’, ‘swim’}
Lem_tokens = {‘A’, ‘swimmer’, ‘like’, ‘swim’, ‘thus’, ‘he’, ‘swim’}

3.2 Multinomial Naïve Bayes model

As discussed earlier Naïve Bayes model is based on the Bayes probability theorem. The probability rule is quite simple. It is as follows

$$\mathrm{P}(w_j|X) = \frac{\mathrm{P}(\mathrm{X}|W_j)\ .\ \mathrm{P}(W_j)}{\mathrm{P}(\mathrm{X})}$$

P(w_j|X) is a posterior probability ,which tells that what is the probability that a particular object belongs to class w_j given its feature values X?

P(X|w_j) is a conditional probability, which gives the probability of finding a features X in class w_j. In Naïve Bayes model conditional probability can be computed by multiplying each feature conditional probability with respect to given class w_j, because of the features mutual independence assumption. That follows like this

$$\mathrm{P}(\mathrm{X}|w_j) = \mathrm{P}(X_1|w_j).\ \mathrm{P}(X_2|w_j).\ \mathrm{P}(X_3|w_j)\ldots..\ \mathrm{P}(X_d|w_j)$$

Where d is the dimension of the feature vector. And then the feature conditional probabilities in Multinomial Naïve Bayes model are calculated considering the term frequency. It is as follows

$$P(X_i \mid w_j) = \frac{\sum tf(X_i, d \in w_j) + \alpha}{\sum N_{d \epsilon w_j} + \alpha . V}$$

Where

- X_i : A word from a feature vector X of a particular sample.
- $\sum tf(X_i, d \in w_j)$: The sum of raw term frequencies of word from all documents in the training sample that belong to class w_j.
- $\sum N_{d \epsilon w_j}$: The sum of all term frequencies in the training dataset for class w_j.
- α: An additive smoothing parameter (for Laplace smoothing = 1).
- V: The size of vocabulary.

P(w_j) is a class probability which tells that general probability of encountering a particular class. It can be calculated as follows

$$P(w_j) = \frac{N_{w_j}}{N_c}$$

Where

N_{w_j} : count of samples from class w_j.

N_c : count of all samples

And finally P(X) tells that probability of encountering a particular pattern X independent from the class label. But this is constant term so we can ignore it while calculating the posterior probability. It can be calculated as follows

$$P(X) = P(X|W_j).P(W_j)+P(X|W_j^{\,c}).P(W_j^{\,c})$$

Chapter 4

NLTK(Natural Language Processing Tool Kit)

Processing of text is an essential part of spam filtering. In python there is an inbuilt text processing tool kit which is Natural Language Processing Tool Kit. It is so rich of many text corpora collected from various places. And also NLTK is composed of so many text processing operations. Following study explains the tokenizer and stemmers of NLTK. And along with them it also explains about enron emain corpus and stop words corpus.

4.1 Enron Email Corpus

A large set of email messages, the Enron corpus, was made public during the legal investigation concerning the Enron corporation. The raw corpus is currently available on the web at http://www-2.cs.cmu.edu/~enron/. The current version contains 619,446 messages belonging to 158 users. This has various kinds of emails from different users that make the algorithms which learn from the features of this corpus to have strong command on different email types. And this corpus has email messages which are already preprocessed for experimental use. Due to the computational power limitation I trained my algorithm using only of 8000 ham and 8000 spam messages.

4.2 Stop words corpus

As mentioned above [16] stop words are the words that are particularly considered common in text corpus and thus considered as rather uninformative. Emails are composed of many stop words, they would not contribute much as features to classifying the new messages. So we need them to be eliminated from the feature set. Two approaches we have discussed about how to find stop words[6]. One is that find the most frequent words in all the documents and by using threshold limit we can eliminate all the words whose frequency in documents is above the threshold. Another approach to follow is that to have set of stop words and search each feature against this set if it is any stop word. The latter approach can be followed using the English stop words corpus of NLTK. It is a rich of all the stop words in English language. This can makes the user work simple to eliminate all the unnecessary features.

4.3 Tokenizers

Tokenization is the task of cutting a string into identifiable linguistic units that constitute a piece of language data. Tokenization of a text is necessary in order to have features. In NLTK there are some tokenizers to make the users work simple. They are very simple to use and efficient in working. Regular expressions are the pattern matching strings. They contain some pattern to split the text. A simple example for regular expression tokenization is splitting the data whenever the space is encountered in the text, here space acts as regular expression pattern.
NLTK has a method called regular expression tokenizer which takes text and regular expression pattern to split the text according to that regular expression.
For example nltk.regexp_tokenize('ram got 100$ for his work.', r'\w+|?$\d+(\.\d+)|[.,]') will split the give sentence into as follows

Token= {'ram', 'got', '100$', 'for', 'his', 'work', '.'}
Where
\w+ : pattern to identify strings of alphabets
\d+ : pattern to identify strings of digits

4.4 Stemmers

Stemming is an effective process in the feature extraction which is used to reduce the redundant features of the feature set. Words are reduced to their meaningful form. NLTK presents Porter stemmer and Lancaster stemmer which are have their own way of stemming the text. Porter stemmer generates more meaningful tokens than Lancaster stemmer. In the following work I have used the porter stemmer to stem the suffixes of the strings.

Chapter 5

Methodology

The aim of this project is to extract the features from the preprocessed emails in enron email corpus. Then use those features to train the classifier algorithm and use then test the performance accury of the classifier using the test emails.

5.1 Feature Extraction

This section describes how features from the preprocessed email corpus can be extracted. And the further processing steps in bag words model.
Steps for feature extraction are:

1. Reading email content
2. Using regular expression to tokenize
3. Unicode encoding of tokens.
4. Removal of stop words
5. Feature length heuristic
6. Feature stemming

Emails collected from the enron corpus are stored in folders train and test to train the classifier and to test it respectively. And these emails can be read in the python using the os class file reading methods. For every email in the train folder a feature vector will be made. For each email In train folder regular expression tokenizer is used to split the content into tokens. Then these tokens will be converted into Unicode format using the Unicode encoding methods. This encoding is useful because the emails are encoded with different encoding formats so converting them into common encoding is needed to apply the operations. Removal of stop words can be done using the NLTK English stop words corpus. Unicode encoded tokens which are stop words will be eliminated.

For collecting most important features here feature length heuristic is used. Using feature length heuristic only features which have a length more than a minimum threshold and less than maximum threshold are collected. Here the minimum threshold length used Is 2 and maximum threshold length is 20. This is very useful heuristic because features which are less than a minimum threshold length will not contribute much to classifying and features with more than a maximum threshold length will be considered as strings of nonsense. It means in spam emails it is common to encounter strings of characters which don’t have any semantic meaning. And as a final process of feature extraction stemming is done using the NLTK porter stemmer.

5.2 Training classifier

Training a classifier is required to make it learn about the spam and ham features. Below algorithm is used to train the classifier. Each feature frequency count is stored in train method. Given classifier object maintains the details of features. To maintain this feature count dictionaries are used to store. There are three dictionaries in the classifier __init__ method. The first feature_count is a dictionary of dictionaries. In that two dictionaries of labels spam and ham are stored as spam and ham, each dictionary will contain the feature count of each feature in a given email label. And label_word_count is a dictionary which is used to store the number of total words in a given label. Total_word_count is a variable to store the total word count in trainin data. Total_email_count is used to store the total number of emails used to train and label_email_count is used to store the number of emails from a given label are used to train. This information is useful in calculating the probabilities of Naïve Bayes algorithm. In train method for each email feature set the above details are added to classifier object.

Algorithm

```
Class classifier(object):
__Init__():
        Feature_count=frequency of feature word in given email label
        Label_word_count = number of words in given email label
        Total_word_count = total feature words
        Total_email_count=total email count
        Total_label_email_count= email count in given label


Train(email_features,label):
For each feature_word in email_features of email:
        Feature_count[label][feature]=feature_count[label][feature]+1
        Label_word_count[label]=label_word_count[label]+1
        Total_word_count=total_word_count+1
Total_email_count=Total_email_count+1
Total_label_count[label]=total_label_count[label]+1
```

5.3 Testing the Classifier

After training a classifier it is required to know how much the classifier has learned. So testing of a classifier is required. In testing we need to have test emails. So for testing emails are used which are stored in test folder, these are different emails from the trained emails. To test emails it is necessary to extract features from those emails. After extracting features these are passed to classify method for classification. In classify method probabilities of features with respect to two labels will be compared and the one with the highest probability label will be returned. These probabilities of features with respect to given label can be calculated as follows, Feature_probability method will calculate the probabilities for individual feature in features set as discussed earlier[7] and Email_probability method calculates the probability of each email feature set occurring in a given label by multiplying the individual feature probabilities of email features. And final probability is calculated by multiplying the email probability with given label probability. After getting the results for all emails classification then accuracy of a classifier is calculated as how many emails classified correctly divides by the total emails in test data.

Algorithm

```
Extract_features(email):
    email_content=email
    tokens = regexp_tokinizer(email_content)
    stop_words=['is','was','that'…etc]
    words=words in tokens which are not present in stop_words
    features=for each word in words stem(word) #ex:stem(eating)=eat return
    features


Feature_probability(classifier,feature_word,label):
$P = \frac{(\text{classifier.feature\_count[label][feature\_word]+alpha })}{\text{calssifier.label\_word\_count[label]}+(\text{alpha}*\text{total\_vocabulary\_count})}$
Return p


Email_probability(classifier,email_features,label):
Email_prob=1.0
For each feature in email_features:
    Email_prob = Email_probability*feature_probability(classifier,feature,label)
Return Email_probabilty
```

```
Probability(classifier,features,label):
    Label_prob = $\frac{\text{classifier.total\_label\_count[label]}}{\text{classifier.total\_email\_count}}$
    Prob=Label_prob*email_probabilty(classifier,features,label)
    Return Prob


Classify(email):
    Spam_prob=Probability(classifier,email,spam)
    Ham_prob= Probability(classifier,email,ham)
    If spam_prob > ham_prob:
        return spam
    Else : return ham


Test(classifer):
Total_email = number of emails in test data
For each email in test data
    result=classify(email)
    If result == email_label:
        correct=correct+1
Accuracy = $\frac{\text{correct}}{\text{total\_email}}$
Return Accuracy
```

Chapter 6

Results and conclusion

Result

All the training and testing was done on public enron corpus containing preprocessed emails from different users. I was trained classifier with 8000 emails from ham and 8000 emails from spam. For testing 200 emails were used, 100 from spam and 100 from ham label. Naïve Bayes classification algorithm was tested for its accuracy. It has given accuracy more than 80%.

Conclusion

Project work I was taken in machine learning has given me a great experience to learn various concepts of machine learning and to gain my knowledge on artificial intelligence techniques.