GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

SPAM CLASSIFICATION USING MACHINE LEARNING

A Report for the Evaluation 3 of Project 2

Submitted by

NAMAN BISHNOI

(1613114023)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION OF COMPUTER NETWORK AND CYBER SECURITY

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Mr. S. PRAKASH

Professor

APRIL / MAY- 2020

GALGOTIAS
UNIVERSITY

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"SPAM CLASSIFICATION USING MACHINE LEARNING"** is the bonafide work of **"NAMAN BISHNOI (1613114057)"** who carried out the project work under my supervision.

SIGNATURE OF HEAD
Dr. MUNISH SHABARWAL,
PhD (Management), PhD (CS)
Professor & Dean,
School of Computing Science & Engineering

SIGNATURE OF SUPERVISOR
Mr. S. PRAKASH
Professor
School of Computing Science & Engineering

Abstract

Spamming is the process of posting unwanted and not related comments on specific posts in any type of social sharing medium or video-sharing medium. These messages are posted by bots for reducing ranking or disturbing users viewing experience which ultimately reduces the rank of website and post. This spamming is done manually also which are mostly seen in most competitive pages.

There are few methods that can remove spamming methods that use data mining techniques but in this project, we are automating the process of spam comment detection using machine learning by taking a dataset of youtube spam messages and applying count vectorizer and naive bayes algorithm for clustering on the given dataset using Go programming language.

Anyone having an e-mail address must have faced unwanted e-mails which we call spam mail. Modern spam filtering software are continuously struggling to detect unwanted e- mails and mark them as spam mail. It is an ongoing battle between spam filtering software and anonymous spam mail senders to defeat each other. Because of that, it is very important to improve spam filters algorithm time to time. Behind the scenes, we use Machine-learning algorithm to find unwanted e-mails. More specifically, we use text classifier algorithm like Naïve Bayes, Support Vector Machine or Neural Network to do the job. In this article, I will try to show you how to use Naïve Bayes algorithm to identify spam e-mail. I will also try to compare the results based on statistics.

The **source code** for this project is available on Github repository

https://github.com/diabloxenon/Spamaway.git

ACKNOWLEDGEMENTS:

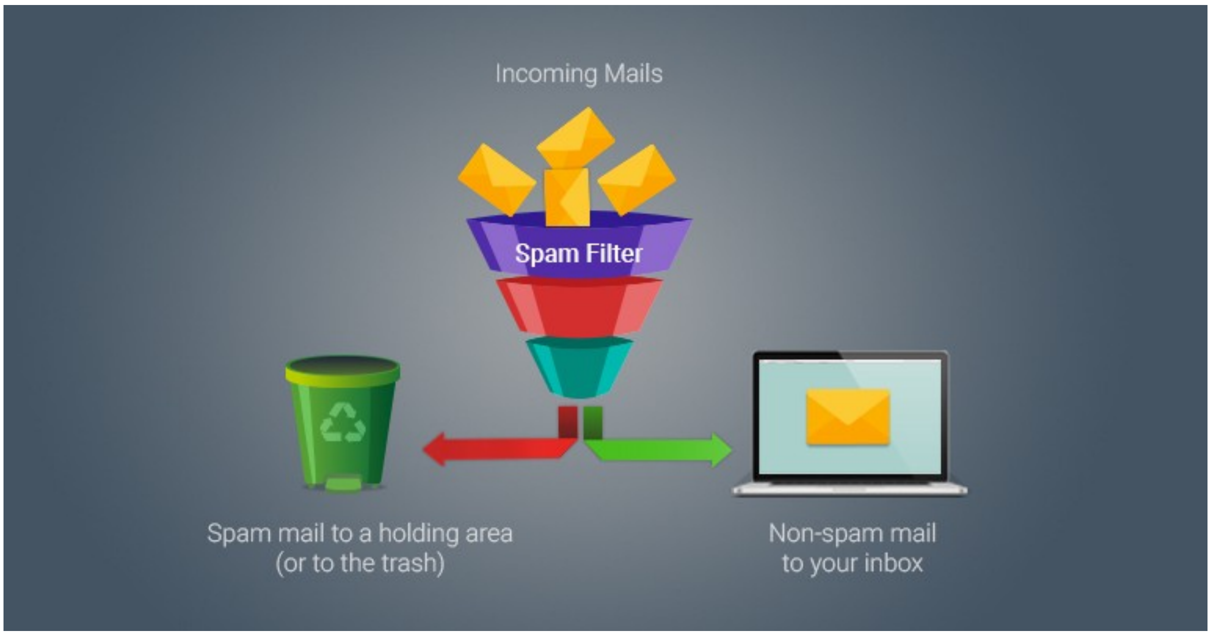
I thank my advisor, Mr. S. Prakash, whose guidance, support, and dedication is priceless. Mr. S. Prakash is an educator in the truest sense of the word. I especially thank to IEEE (Institute of Electrical and Electronics Engineers, Inc.) for providing recent information about my topic.

It has been a challenging, yet rewarding journey which I could not have completed alone and I am grateful for your support.

Thank you.

TABLES OF CONTENT

Introduction


Incoming Mails
Spam Filter
Spam mail to a holding area (or to the trash)
Non-spam mail to your inbox

This project, count vectorizer is used for extracting features form a given dataset and design model by generating tests and training sets from given data. Then the naive bayes classifier is applied for clustering and the test and training set is given as input based on this data given message is tested if it is spam or not. The idea of automatically classifying spam and non-spam emails by applying machine learning methods has been popular in academia and has been a topic of interest for many researchers.

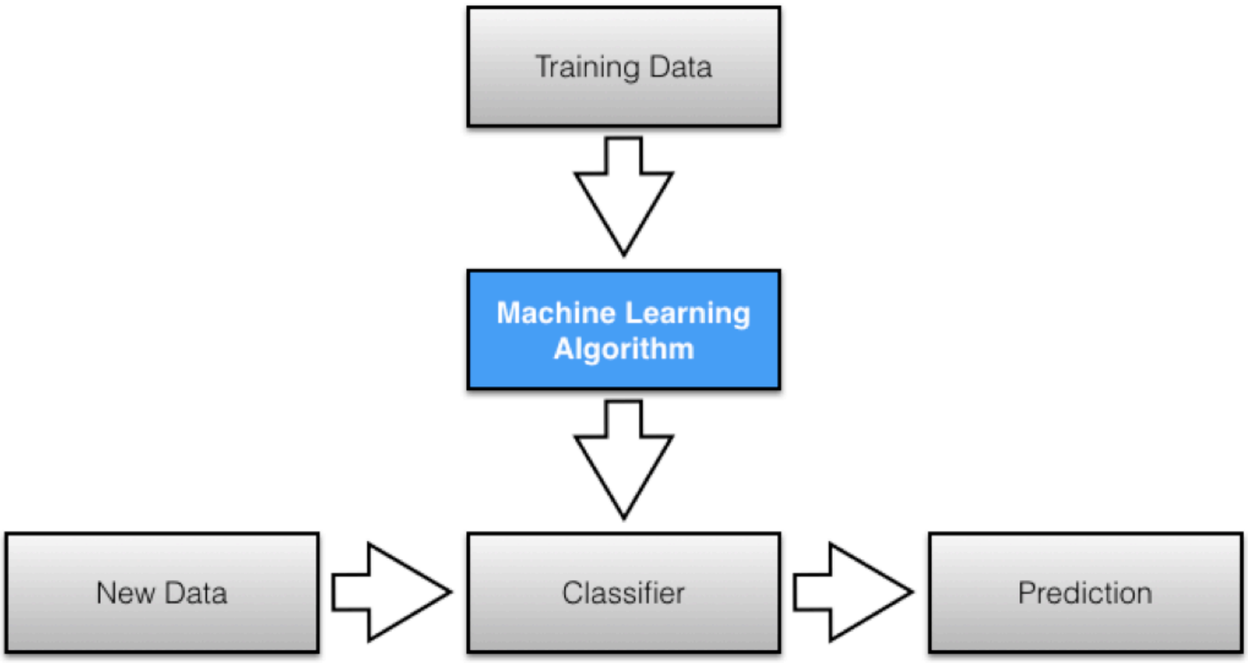
Knowledge engineering and machine learning are the two main approaches scientists have applied to overcome the spam-filtering problem. The first solution focuses on creating a knowledge-based system in which pre-defined rules dictate if an incoming message is legitimate or not. The primary disadvantage of this method is that those rules need to be maintained and updated continuously by the user or a 3rd party like for example a software vendor.

In the existing system, data mining techniques are used for detecting spam messages. Most of these methods work only after posting messages. There is a need for a system that can automate this process before posting message. The spam detection problem is in fact a text classification problem. An e-mail (a text document) is either “spam” or “no spam”. In text mining, this is called single-label text classification, since there is only one label: “spam”. A

classifier is an algorithm that is capable of telling whether a text document is either “spam” or “no spam”.

Machine Learning and Data Mining

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks.



Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Statistical learning in language acquisition

Statistical learning is the ability for humans and other animals to extract statistical regularities from the world around them to learn about the environment. Although statistical learning is now thought to be a generalized learning mechanism, the phenomenon was first identified in human infant language acquisition.

The earliest evidence for these statistical learning abilities comes from a study by Jenny Saffran, Richard Aslin, and Elissa Newport, in which 8-month-old infants were presented with nonsense streams of monotone speech. Each stream was composed of four three-syllable “pseudowords” that were repeated randomly. After exposure to the speech streams for two minutes, infants reacted differently to hearing “pseudowords” as opposed to “nonwords” from the speech stream, where nonwords were composed of the same syllables that the infants had been exposed to, but in a different order. This suggests that infants are able to learn statistical relationships between syllables even with very limited exposure to a language. That is, infants learn which syllables are always paired together and which ones only occur together relatively rarely, suggesting that they are parts of two different units. This method of learning is thought to be one way that children learn which groups of syllables form individual words.

Since the initial discovery of the role of statistical learning in lexical acquisition, the same mechanism has been proposed for elements of phonological acquisition, and syntactical acquisition, as well as in non-linguistic domains. Further research has also indicated that statistical learning is likely a domain-general and even species-general learning mechanism, occurring for visual as well as auditory information, and in both primates and non-primates.

Lexical Acquisition

The role of statistical learning in language acquisition has been particularly well documented in the area of lexical acquisition. One important contribution to infants' understanding of segmenting words from a continuous stream of speech is their ability to recognize statistical regularities of the speech heard in their environments. Although many factors play an important role, this specific mechanism is powerful and can operate over a short time scale.

Original Findings

It is a well-established finding that, unlike written language, spoken language does not have any clear boundaries between words; spoken language is a continuous stream of sound rather than individual words with silences between them. This lack of segmentation between linguistic units presents a problem for young children learning language, who must be able to pick out individual units from the continuous speech streams that they hear. One proposed method of how children are able to solve this problem is that they are attentive to the statistical regularities of the world around them. For example, in the phrase "pretty baby," children are more likely to hear the sounds *pre* and *ty* heard together during the entirety of the lexical input around them than they are to hear the sounds *ty* and *ba* together. In an artificial grammar learning study with adult participants, Saffran, Newport, and Aslin found that participants were able to locate word boundaries based only on transitional probabilities, suggesting that adults are capable of using statistical regularities in a language-learning task. This is a robust finding that has been widely replicated.

To determine if young children have these same abilities Saffran Aslin and Newport exposed 8-month-old infants to an artificial grammar. The grammar was composed of four words, each composed of three nonsense syllables. During the experiment, infants heard a continuous speech stream of these words . Importantly, the speech was presented in a monotone with no cues (such as pauses, intonation, etc.) to word boundaries other than the statistical probabilities. Within a word, the transitional probability of two syllable pairs was 1.0: in the word *bidaku*, for example, the probability of hearing the syllable *da* immediately after the syllable *bi* was 100%. Between words, however, the transitional probability of hearing a syllable pair was much lower: After any given word (e.g., *bidaku*) was presented, one of three words could follow (in this case, *padoti*, *golabu*, or *tupiro*), so the likelihood of hearing any given syllable after *ku* was only 33%.

To determine if infants were picking up on the statistical information, each infant was presented with multiple presentations of either a word from the artificial grammar or a nonword made up of the same syllables but presented in a random order. Infants who were presented with nonwords during the test phase listened significantly longer to these words than infants who were presented with words from the artificial grammar, showing a novelty preference for these new nonwords. However, the implementation of the test could also be due to infants learning serial-order information and not to actually learning transitional probabilities between words. That is, at test, infants heard strings such as *dapiku* and *tilado* that were never presented during learning; they could simply have learned that the syllable *ku* never followed the syllable *pi*.

To look more closely at this issue, Saffran Aslin and Newport conducted another study in which infants underwent the same training with the artificial grammar but then were presented with either words or part-words rather than words or nonwords. The part-words were syllable sequences composed of the last syllable from one word and the first two syllables from another (such as *kupado*). Because the part-words had been heard during the time when children were listening to the artificial grammar, preferential listening to these part-words would indicate that children were learning not only serial-order information, but also the statistical likelihood of hearing particular syllable sequences. Again, infants showed greater listening times to the novel (part-) words, indicating that 8-month-old infants were able to extract these statistical regularities from a continuous speech stream.

Further Research

This result has been the impetus for much more research on the role of statistical learning in lexical acquisition and other areas (see). In a follow-up to the original report, Aslin, Saffran, and Newport found that even when words and part words occurred equally often in the speech stream, but with different transitional probabilities between syllables of words and part words, infants were still able to detect the statistical regularities and still preferred to listen to the novel part-words over the familiarized words. This finding provides stronger evidence that infants are able to pick up transitional probabilities from the speech they hear, rather than just being aware of frequencies of individual syllable sequences.

Another follow-up study examined the extent to which the statistical information learned during this type of artificial grammar learning feeds into knowledge that infants may already have about their native language. Infants preferred to listen to words over part-words, whereas there was no significant difference in the nonsense frame condition. This finding suggests that even pre-linguistic infants are able to integrate the statistical cues they learn in a laboratory into their previously-acquired knowledge of a language. In other words, once infants have acquired some linguistic knowledge, they incorporate newly acquired information into that previously-acquired learning.

A related finding indicates that slightly older infants can acquire both lexical and grammatical regularities from a single set of input, suggesting that they are able to use outputs of one type of statistical learning (cues that lead to the discovery of word boundaries) as input to a second type (cues that lead to the discovery of syntactical regularities. At test, 12-month-olds preferred to listen to sentences that had the same grammatical structure as the artificial language they had been tested on rather than sentences that had a different (ungrammatical) structure. Because learning grammatical regularities requires infants to be able to determine boundaries between individual words, this indicates that infants who are still quite young are able to acquire multiple levels of language knowledge (both lexical and syntactical) simultaneously, indicating that statistical learning is a powerful mechanism at play in language learning.

Despite the large role that statistical learning appears to play in lexical acquisition, it is likely not the only mechanism by which infants learn to segment words. Statistical learning studies are generally conducted with artificial grammars that have no cues to word boundary information other than transitional probabilities between words. Real speech, though, has many different types of cues to word boundaries, including prosodic and phonotactic information.

Together, the findings from these studies of statistical learning in language acquisition indicate that statistical properties of the language are a strong cue in helping infants learn their first language.

Phonological Acquisition

There is much evidence that statistical learning is an important component of both discovering which phonemes are important for a given language and which contrasts within phonemes are important. Having this knowledge is important for aspects of both speech perception and speech production.

Distributional Learning

Since the discovery of infants' statistical learning abilities in word learning, the same general mechanism has also been studied in other facets of language learning. For example, it is well-established that infants can discriminate between phonemes of many different languages but eventually become unable to discriminate between phonemes that do not appear in their native language; however, it was not clear how this decrease in discriminatory ability came about. Maye et al. suggested that the mechanism responsible might be a statistical learning mechanism in which infants track the distributional regularities of the sounds in their native language. To test this idea, Maye et al. exposed 6- and 8-month-old infants to a continuum of speech sounds that varied on the degree to which they were voiced. The distribution that the infants heard was either bimodal, with sounds from both ends of the voicing continuum heard most often, or unimodal, with sounds from the middle of the distribution heard most often. The results indicated that infants from both age groups were sensitive to the distribution of phonemes. At test, infants heard either non-alternating (repeated exemplars of tokens 3 or 6 from an 8-token continuum) or alternating (exemplars of tokens 1 and 8) exposures to specific phonemes on the continuum. Infants exposed to the bimodal distribution listened longer to the alternating trials than the non-alternating trials while there was no difference in listening times for infants exposed to the unimodal distribution. This finding indicates that infants exposed the bimodal distribution were better able to discriminate sounds from the two ends of the distribution than were infants in the unimodal condition, regardless of age. This type of statistical learning differs from that used in lexical acquisition, as it requires infants to track frequencies rather than transitional probabilities, and has been named "distributional learning."

Distributional learning has also been found to help infants contrast two phonemes that they initially have difficulty in discriminating between. Maye, Weiss, and Aslin found that infants who were exposed to a bimodal distribution of a non-native contrast that was initially difficult to discriminate were better able to discriminate the contrast than infants exposed to a unimodal distribution of the same contrast. Maye et al. also found that infants were able to abstract features of a contrast (i.e., voicing onset time) and generalize that feature to the same type of contrast at a different place of articulation, a finding that has not been found in adults.

In a review of the role of distributional learning on phonological acquisition, Werker et al. note that distributional learning cannot be the only mechanism by which phonetic categories are acquired. However, it does seem clear that this type of statistical learning mechanism can play a role in this skill, although research is ongoing.

Perceptual Magnet Effect

A related finding regarding statistical cues to phonological acquisition is a phenomenon known as the perceptual magnet effect. In this effect, a prototypical phoneme of a person's native language acts as a “magnet” for similar phonemes, which are perceived as belonging to the same category as the prototypical phoneme. In the original test of this effect, adult participants were asked to indicate if a given exemplar of a particular phoneme differed from a referent phoneme. If the referent phoneme is a non-prototypical phoneme for that language, both adults and 6-month-old infants show less generalization to other sounds than they do for prototypical phonemes, even if the subjective distance between the sounds is the same. That is, adults and infants are both more likely to notice that a particular phoneme differs from the referent phoneme if that referent phoneme is a non-prototypical exemplar than if it is a prototypical exemplar. The prototypes themselves are apparently discovered through a distributional learning process, in which infants are sensitive to the frequencies with which certain sounds occur and treat those that occur most often as the prototypical phonemes of their language.

Syntactical Acquisition

A statistical learning device has also been proposed as a component of syntactical acquisition for young children.[1][9][17] Early evidence for this mechanism came largely from studies of computer modeling or analyses of natural language corpora.[18][19] These early studies focused largely on distributional information specifically rather than statistical learning mechanisms generally. Specifically, in these early papers it was proposed that children created templates of possible sentence structures involving unnamed categories of word types (i.e., nouns or verbs, although children would not put these labels on their categories). Children were thought to learn which words belonged to the same categories by tracking the similar contexts in which words of the same category appeared.

Later studies expanded these results by looking at the actual behavior of children or adults who had been exposed to artificial grammars.[9] These later studies also considered the role of statistical learning more broadly than the earlier studies, placing their results in the context of the statistical learning mechanisms thought to be involved with other aspects of language learning, such as lexical acquisition.

Experimental Results

Evidence from a series of four experiments conducted by Gomez and Gerken suggests that children are able to generalize grammatical structures with less than two minutes of exposure to an artificial grammar.[9][20] In the first experiment, 11-12 month-old infants were trained on an artificial grammar composed of nonsense words with a set grammatical structure. At test, infants heard both novel grammatical and ungrammatical sentences. Infants oriented longer towards the grammatical sentences, in line with previous research that suggests that infants generally orient for a longer amount of time to natural instances of language rather than altered instances of language e.g.,.[21] (This familiarity preference differs from the novelty preference generally found in word-learning studies, due to the differences between lexical acquisition and syntactical acquisition.) This finding indicates that young children are sensitive to the grammatical structure of language even after minimal exposure. Gomez and Gerken also found that this sensitivity is evident when ungrammatical transitions are located in the middle of the sentence (unlike in the first experiment, in which all the errors occurred at the beginning and end of the sentences), that the results could not be due to an innate preference for the grammatical sentences caused by something other than grammar, and that children are able to generalize the grammatical rules to new vocabulary.

Together these studies suggest that infants are able to extract a substantial amount of syntactic knowledge even from limited exposure to a language.[9][20] Children apparently detected grammatical anomalies whether the grammatical violation in the test sentences occurred at the end or in the middle of the sentence. Additionally, even when the individual words of the grammar were changed, infants were still able to discriminate between grammatical and ungrammatical strings during the test phase. This generalization indicates that infants were not learning vocabulary-specific grammatical structures, but abstracting the general rules of that grammar and applying those rules to novel vocabulary. Furthermore, in all four experiments, the test of grammatical structures occurred five minutes after the initial exposure to the artificial grammar had ended, suggesting that the infants were able to maintain the grammatical abstractions they had learned even after a short delay.

In a similar study, Saffran found that adults and older children (first and second grade children) were also sensitive to syntactical information after exposure to an artificial language which had no cues to phrase structure other than the statistical regularities that were present.[22] Both adults and children were able to pick out sentences that were ungrammatical at a rate greater than chance, even under an “incidental” exposure condition in which participants’ primary goal was to complete a different task while hearing the language.

Although the number of studies dealing with statistical learning of syntactical information is limited, the available evidence does indicate that the statistical learning mechanisms are likely a contributing factor to children's ability to learn their language.

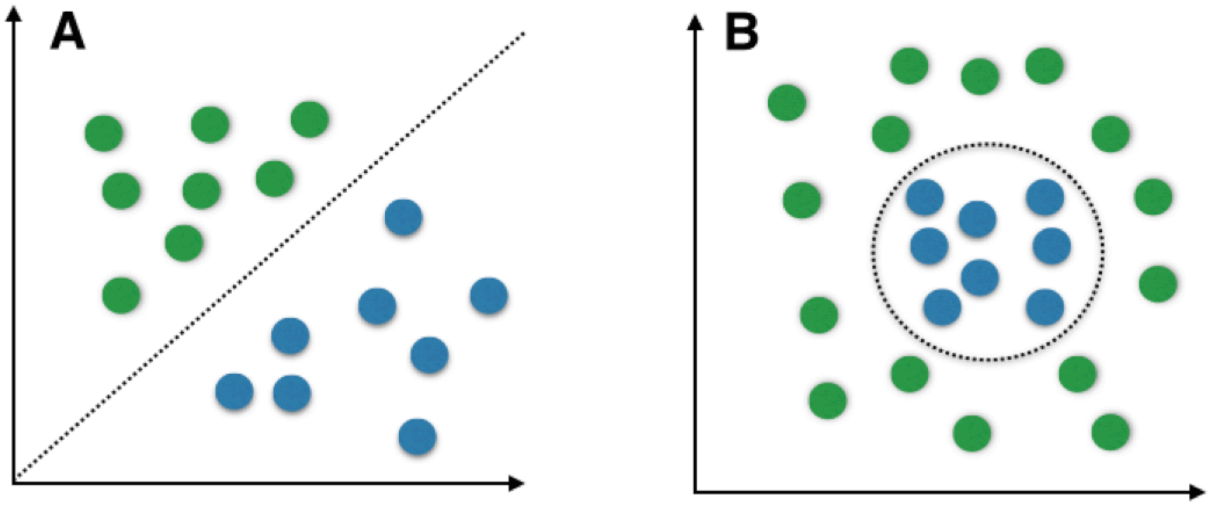
Statistical Learning in Bilingualism

Much of the early work using statistical learning paradigms focused on the ability for children or adults to learn a single language,[1] consistent with the process of language acquisition for monolingual speakers or learners. However, it is estimated that approximately 60-75% of people in the world are bilingual.[23] More recently, researchers have begun looking at the role of statistical learning for those who speak more than one language. Although there are no reviews on this topic yet, Weiss, Gerfen, and Mitchel examined how hearing input from multiple artificial languages simultaneously can affect the ability to learn either or both languages.[24] Over four experiments, Weiss et al. found that, after exposure to two artificial languages, adult learners are capable of determining word boundaries in both languages when each language is spoken by a different speaker. However, when the two languages were spoken by the same speaker, participants were able learn both languages only when they were "congruent"—when the word boundaries of one language matched the word boundaries of the other. When the languages were incongruent—a syllable that appeared in the middle of a word in one language appeared at the end of the word in the other language—and spoken by a single speaker, participants were able to learn, at best, one of the two languages. A final experiment showed that the inability to learn incongruent languages spoken in the same voice was not due to syllable overlap between the languages but due to differing word boundaries.

Similar work replicates the finding that learners are able to learn two sets of statistical representations when an additional cue is present (two different male voices in this case).[25] In their paradigm, the two languages were presented consecutively, rather than interleaved as in Weiss et al.'s paradigm,[24] and participants did learn the first artificial language to which they had been exposed better than the second, although participants' performance was above chance for both languages.

While statistical learning improves and strengthens multilingualism, it appears that the inverse is not true. In a study by Yim and Rudoy[26] it was found that both monolingual and bilingual children perform statistical learning tasks equally well.

Naive bayes



Short introduction to Bayes Theorem

In machine learning we are often interested in selecting the best hypothesis (h) given data (d).

In a classification problem, our hypothesis (h) may be the class to assign for a new data instance (d).

One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge.

Bayes' Theorem is stated as:

$$P(h|d) = (P(d|h) * P(h)) / P(d)$$

Where

- P(h|d) is the probability of hypothesis h given the data d. This is called the posterior probability.
- P(d|h) is the probability of data d given that the hypothesis h was true.
- P(h) is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
- P(d) is the probability of the data (regardless of the hypothesis).

You can see that we are interested in calculating the posterior probability of P(h|d) from the prior probability p(h) with P(D) and P(d|h).

After calculating the posterior probability for a number of different hypotheses, you can select the hypothesis with the highest probability. This is the maximum probable hypothesis and may formally be called the maximum a posteriori (MAP) hypothesis.

This can be written as:

$$MAP(h) = max(P(h|d))$$

or

$$MAP(h) = max((P(d|h) * P(h)) / P(d))$$

or

$$MAP(h) = max(P(d|h) * P(h))$$

The P(d) is a normalizing term which allows us to calculate the probability. We can drop it when we are interested in the most probable hypothesis as it is constant and only used to normalize.

Back to classification, if we have an even number of instances in each class in our training data, then the probability of each class (e.g. P(h)) will be equal. Again, this would be a constant term in our equation and we could drop it so that we end up with:

$$MAP(h) = max(P(d|h))$$

Naive Bayes Classification

In machine learning, **naïve Bayes classifiers** are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models.[1] But they could be coupled with Kernel density estimation and achieve higher accuracy levels.[2][3]

Naïve Bayes has been studied extensively since the 1960s. It was introduced (though not under that name) into the text retrieval community in the early 1960s,[4] and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (document categorization)(such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines.[5] It also finds application in automatic medical diagnosis.[6]

Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression,[7]:718 which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, naive Bayes models are known under a variety of names, including **simple Bayes** and **independence Bayes**.[8] All these names reference the use of Bayes' theorem in the classifier's decision rule, but naïve Bayes is not (necessarily) a Bayesian method.

Introduction

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers.[9] Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.[10]

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

Probabilistic model

Training is fast because only the probability of each class and the probability of each class given different input (x) values need to be calculated. No coefficients need to be fitted by optimization procedures.

Calculating Class Probabilities

The class probabilities are simply the frequency of instances that belong to each class divided by the total number of instances.

For example in a binary classification the probability of an instance belonging to class 1 would be calculated as:

P(class=1) = count(class=1) / (count(class=0) + count(class=1))

In the simplest case each class would have the probability of 0.5 or 50% for a binary classification problem with the same number of instances in each class.

Calculating Conditional Probabilities

The conditional probabilities are the frequency of each attribute value for a given class value divided by the frequency of instances with that class value.

For example, if a "*weather*" attribute had the values "*sunny*" and "*rainy*" and the class attribute had the class values "*go-out*" and "*stay-home*", then the conditional probabilities of each weather value for each class value could be calculated as:

- P(weather=sunny | class=go-out) = count(instances with weather=sunny and class=go-out) / count(instances with class=go-out)
- P(weather=sunny | class=stay-home) = count(instances with weather=sunny and class=stay-home) / count(instances with class=stay-home)
- P(weather=rainy | class=go-out) = count(instances with weather=rainy and class=go-out) / count(instances with class=go-out)
- P(weather=rainy | class=stay-home) = count(instances with weather=rainy and class=stay-home) / count(instances with class=stay-home)

Make Predictions With a Naive Bayes Model

Given a naive Bayes model, you can make predictions for new data using Bayes theorem.

*MAP(h) = max(P(d|h) * P(h))*

Using our example above, if we had a new instance with the *weather* of *sunny*, we can calculate:

go-out = P(weather=sunny | class=go-out) * P(class=go-out)

stay-home = P(weather=sunny | class=stay-home) * P(class=stay-home)

We can choose the class that has the largest calculated value. We can turn these values into probabilities by normalizing them as follows:

P(go-out | weather=sunny) = go-out / (go-out + stay-home)

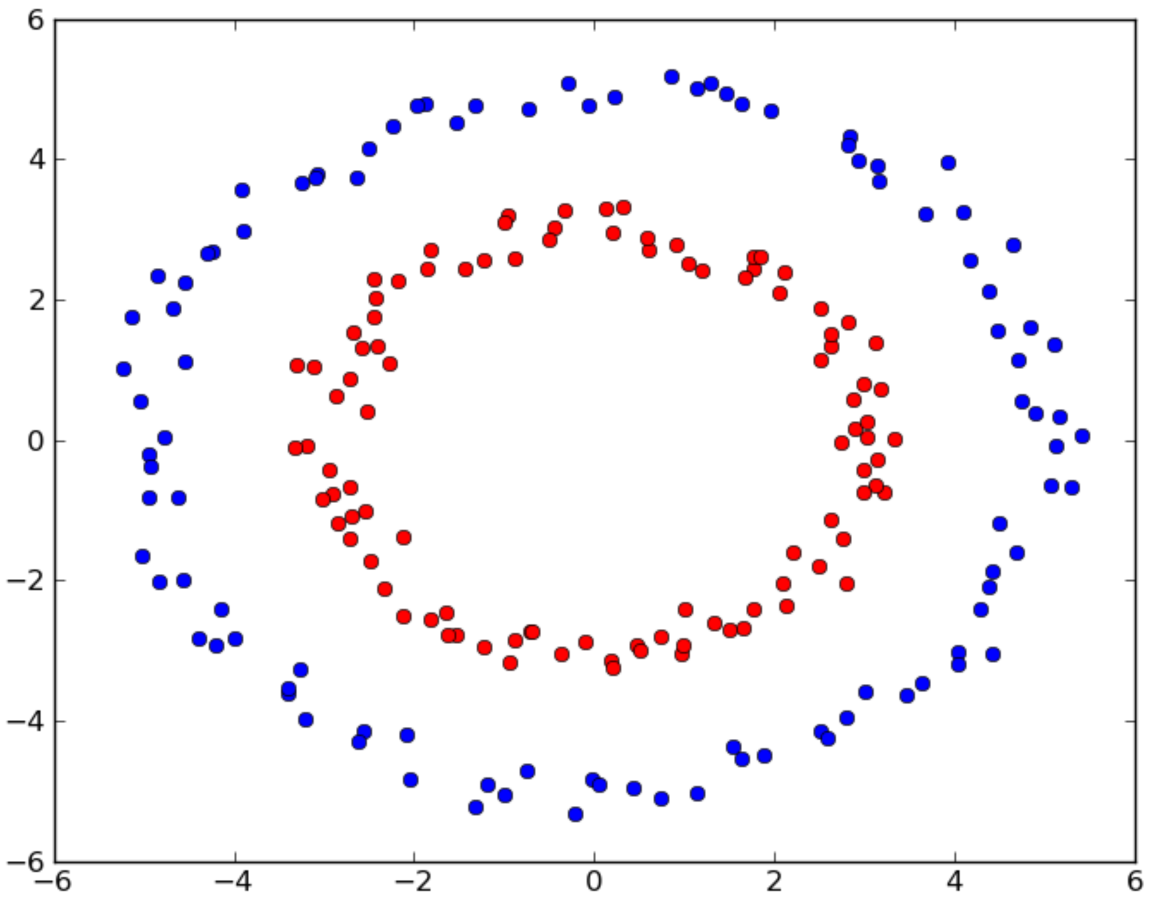
P(stay-home | weather=sunny) = stay-home / (go-out + stay-home)

If we had more input variables we could extend the above example. For example, pretend we have a "*car*" attribute with the values "*working*" and "*broken*". We can multiply this probability into the equation.

For example below is the calculation for the "go-out" class label with the addition of the car input variable set to "working":

go-out = P(weather=sunny | class=go-out) * P(car=working | class=go-out) * P(class=go-out)

Naive Bayes Spam Filtering



Naive Bayes classifiers are a popular statistical technique of e-mail filtering. They typically use bag of words features to identify spam e-mail, an approach commonly used in text classification.

Naive Bayes classifiers work by correlating the use of tokens (typically words, or sometimes other things), with spam and non-spam e-mails and then using Bayes' theorem to calculate a probability that an email is or is not spam.

Naive Bayes spam filtering is a baseline technique for dealing with spam that can tailor itself to the email needs of individual users and give low false positive spam detection rates that are generally acceptable to users. It is one of the oldest ways of doing spam filtering, with roots in the 1990s.

Bayesian algorithms were used to sort and filter email by 1996. Although naive Bayesian filters did not become popular until later, multiple programs were released in 1998 to address the growing problem of unwanted email.[1] The first scholarly publication on Bayesian spam filtering was by Sahami et al. in 1998.[2] That work was soon thereafter deployed in commercial spam filters.[*citation needed*] However, in 2002 Paul Graham greatly decreased the false positive rate, so that it could be used on its own as a single spam filter.[3][4]

Variants of the basic technique have been implemented in a number of research works and commercial software products.[5] Many modern mail clients implement Bayesian spam filtering. Users can also install separate email filtering programs. Server-side email filters, such as DSPAM, SpamAssassin,[6] SpamBayes,[7] Bogofilter and ASSP, make use of Bayesian spam filtering techniques, and the functionality is sometimes embedded within mail server software itself. CRM114, oft cited as a Bayesian filter, is not intended to use a Bayes filter in production, but includes the ″unigram″ feature for reference.[8]

Process

Particular words have particular probabilities of occurring in spam email and in legitimate email. For instance, most email users will frequently encounter the word "Viagra" in spam email, but will seldom see it in other email. The filter doesn't know these probabilities in advance, and must first be trained so it can build them up. To train the filter, the user must manually indicate whether a new email is spam or not. For all words in each training email, the filter will adjust the probabilities that each word will appear in spam or legitimate email in its database. For instance, Bayesian spam filters will typically have learned a very high spam probability for the words "Viagra" and "refinance", but a very low spam probability for words seen only in legitimate email, such as the names of friends and family members.

After training, the word probabilities (also known as likelihood functions) are used to compute the probability that an email with a particular set of words in it belongs to either category. Each word in the email contributes to the email's spam probability, or only the most interesting words. This contribution is called the posterior probability and is computed using Bayes' theorem. Then, the email's spam probability is computed over all words in the email, and if the total exceeds a certain threshold (say 95%), the filter will mark the email as a spam.

As in any other spam filtering technique, email marked as spam can then be automatically moved to a "Junk" email folder, or even deleted outright. Some software implement quarantine mechanisms that define a time frame during which the user is allowed to review the software's decision.

The initial training can usually be refined when wrong judgements from the software are identified (false positives or false negatives). That allows the software to dynamically adapt to the ever-evolving nature of spam.

Some spam filters combine the results of both Bayesian spam filtering and other heuristics (pre-defined rules about the contents, looking at the message's envelope, etc.), resulting in even higher filtering accuracy, sometimes at the cost of adaptiveness.

Mathematical Foundation

Bayesian email filters utilize Bayes' theorem. Bayes' theorem is used several times in the context of spam:

- a first time, to compute the probability that the message is spam, knowing that a given word appears in this message;
- a second time, to compute the probability that the message is spam, taking into consideration all of its words (or a relevant subset of them);
- sometimes a third time, to deal with rare words.

Computing the probability that a message containing a given word is spam

Let's suppose the suspected message contains the word "replica". Most people who are used to receiving e-mail know that this message is likely to be spam, more precisely a proposal to sell counterfeit copies of well-known brands of watches. The spam detection software, however, does not "know" such facts; all it can do is compute probabilities.

The formula used by the software to determine that, is derived from Bayes' theorem.

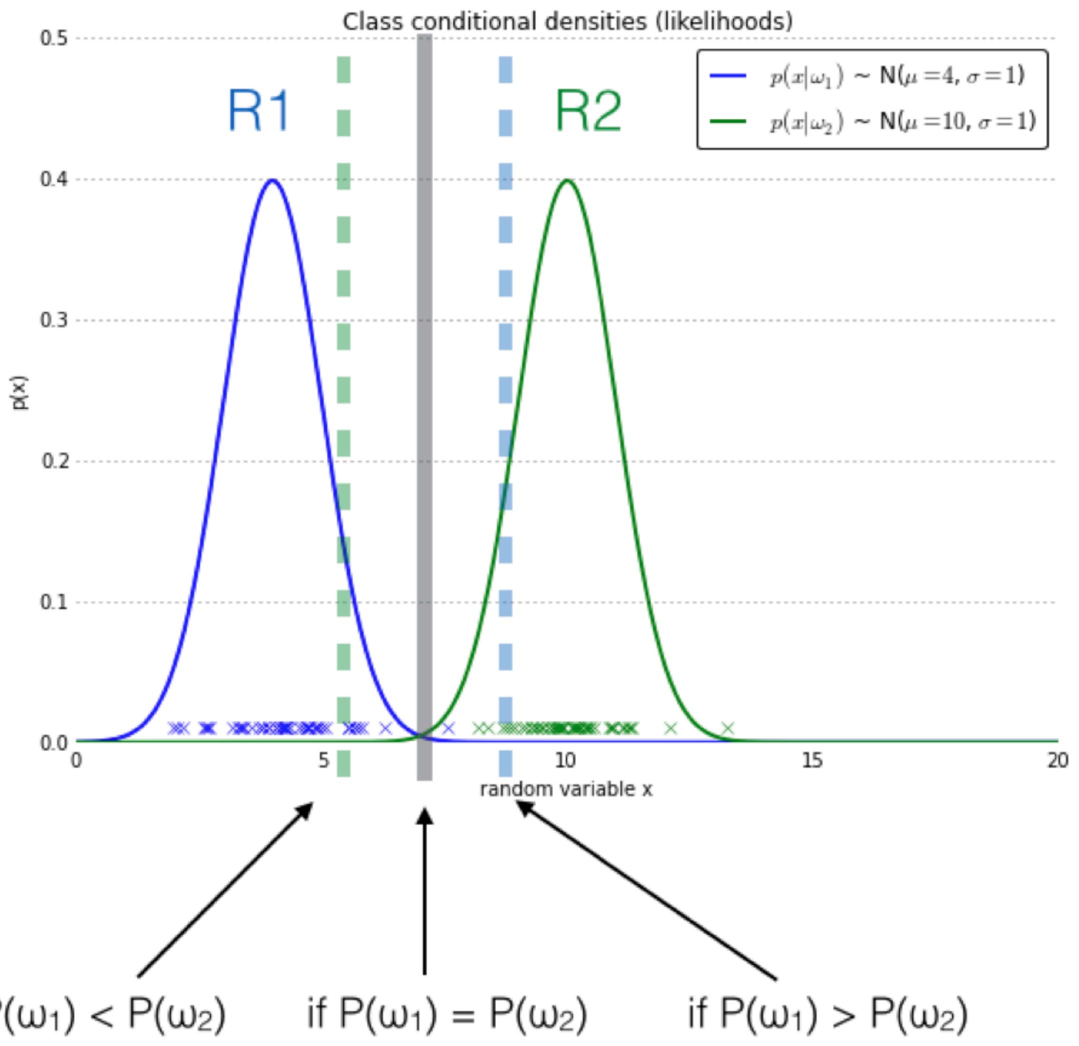
$$\Pr(S|W) = \frac{\Pr(W|S) \cdot \Pr(S)}{\Pr(W|S) \cdot \Pr(S) + \Pr(W|H) \cdot \Pr(H)}$$

where:

- $\Pr(S|W)$ is the probability that a message is a spam, knowing that the word "replica" is in it;
- $\Pr(S)$ is the overall probability that any given message is spam;
- $\Pr(W|S)$ is the probability that the word "replica" appears in spam messages;
- $\Pr(H)$ is the overall probability that any given message is not spam (is "ham");

- Pr(W|H) is the probability that the word "replica" appears in ham messages.

The spamliness of a word



Statistics[9] show that the current probability of any message being spam is 80%, at the very least:

$$\Pr(S) = 0.8; \Pr(H) = 0.2$$

However, most bayesian spam detection software makes the assumption that there is no *a priori* reason for any incoming message to be spam rather than ham, and considers both cases to have equal probabilities of 50%

$$\Pr(S) = 0.5; \Pr(H) = 0.5$$

The filters that use this hypothesis are said to be "not biased", meaning that they have no prejudice regarding the incoming email. This assumption permits simplifying the general formula to:

$$\Pr(S|W) = \frac{\Pr(W|S)}{\Pr(W|S) + \Pr(W|H)}$$

This is functionally equivalent to asking, "what percentage of occurrences of the word "replica" appear in spam messages?"

This quantity is called "spamicity" (or "spaminess") of the word "replica", and can be computed. The number Pr(W|S) used in this formula is approximated to the frequency of messages containing "replica" in the messages identified as spam during the learning phase. Similarly, Pr(W|H) is approximated to the frequency of messages containing "replica" in the messages identified as ham during the learning phase. For these approximations to make sense, the set of learned messages needs to be big and representative enough. It is also advisable that the learned set of messages conforms to the 50% hypothesis about repartition between spam and ham, i.e. that the datasets of spam and ham are of same size.

Of course, determining whether a message is spam or ham based only on the presence of the word "replica" is error-prone, which is why bayesian spam software tries to consider several words and combine their spamicities to determine a message's overall probability of being spam.

Dealing with rare words

In the case a word has never been met during the learning phase, both the numerator and the denominator are equal to zero, both in the general formula and in the spamicity formula. The software can decide to discard such words for which there is no information available.

More generally, the words that were encountered only a few times during the learning phase cause a problem, because it would be an error to trust blindly the information they provide. A simple solution is to simply avoid taking such unreliable words into account as well.

Applying again Bayes' theorem, and assuming the classification between spam and ham of the emails containing a given word ("replica") is a random variable with beta distribution, some programs decide to use a corrected probability:

$$\Pr'(S|W) = \frac{s \cdot \Pr(S) + n \cdot \Pr(S|W)}{s + n}$$

where:

- $\Pr'(S|W)$ is the corrected probability for the message to be spam, knowing that it contains a given word ;
- s is the *strength* we give to background information about incoming spam ;
- $\Pr(S)$ is the probability of any incoming message to be spam ;
- n is the number of occurrences of this word during the learning phase ;
- $\Pr(S|W)$ is the spamicity of this word.

This corrected probability is used instead of the spamicity in the combining formula. Pr(S) can again be taken equal to 0.5, to avoid being too suspicious about incoming email. 3 is a good value for *s*, meaning that the learned corpus must contain more than 3 messages with that word to put more confidence in the spamicity value than in the default value.

This formula can be extended to the case where *n* is equal to zero (and where the spamicity is not defined), and evaluates in this case to Pr(S).

Discussion

Advantages

One of the main advantages[citation needed] of Bayesian spam filtering is that it can be trained on a per-user basis.

The spam that a user receives is often related to the online user's activities. For example, a user may have been subscribed to an online newsletter that the user considers to be spam. This online newsletter is likely to contain words that are common to all newsletters, such as the name of the newsletter and its originating email address. A Bayesian spam filter will eventually assign a higher probability based on the user's specific patterns.

The legitimate e-mails a user receives will tend to be different. For example, in a corporate environment, the company name and the names of clients or customers will be mentioned often. The filter will assign a lower spam probability to emails containing those names.

The word probabilities are unique to each user and can evolve over time with corrective training whenever the filter incorrectly classifies an email. As a result, Bayesian spam filtering accuracy after training is often superior to pre-defined rules.

It can perform particularly well in avoiding false positives,[citation needed] where legitimate email is incorrectly classified as spam. For example, if the email contains the word "Nigeria", which is frequently used in Advance fee fraud spam, a pre-defined rules filter might reject it outright. A Bayesian filter would mark the word "Nigeria" as a probable spam word, but would take into account other important words that usually indicate legitimate e-mail. For example, the name of a spouse may strongly indicate the e-mail is not spam, which could overcome the use of the word "Nigeria."

Disadvantages

Depending on the implementation, Bayesian spam filtering may be susceptible to Bayesian poisoning, a technique used by spammers in an attempt to degrade the effectiveness of spam filters that rely on Bayesian filtering. A spammer practicing Bayesian poisoning will send out emails with large amounts of legitimate text (gathered from legitimate news or literary sources). Spammer tactics include insertion of random innocuous words that are not normally associated with spam, thereby decreasing the email's spam score, making it more likely to slip past a Bayesian spam filter. However, with (for example) Paul Graham's scheme only the most significant probabilities are used, so that padding the text out with non-spam-related words does not affect the detection probability significantly.

Words that normally appear in large quantities in spam may also be transformed by spammers. For example, «Viagra» would be replaced with «Viaagra» or «V!agra» in the spam message. The recipient of the message can still read the changed words, but each of these words is met more rarely by the Bayesian filter, which hinders its learning process. As a general rule, this spamming technique does not work very well, because the derived words end up recognized by the filter just like the normal ones.[17]

Another technique used to try to defeat Bayesian spam filters is to replace text with pictures, either directly included or linked. The whole text of the message, or some part of it, is replaced with a picture where the same text is "drawn". The spam filter is usually unable to analyze this picture, which would contain the sensitive words like «Viagra». However, since many mail clients disable the display of linked pictures for security reasons, the spammer sending links to distant pictures might reach fewer targets. Also, a picture's size in bytes is bigger than the equivalent text's size, so the spammer needs more bandwidth to send messages directly including pictures. Some filters are more inclined to decide that a message is spam if it has mostly graphical contents. A solution used by Google in its Gmail email system is to perform an OCR (Optical Character Recognition) on every mid to large size image, analyzing the text inside.

E-mail Spam Classification

Description

The dataset included for this project is based on a subset of the SpamAssassin Public Corpus. Upper image of Figure 1 shows a sample email that contains a URL, an email address (at the end), numbers, and dollar amounts. While many emails would contain similar types of entities (e.g., numbers, other URLs, or other email addresses), the specific entities (e.g., the specific URL or specific dollar amount) will be different in almost every email. Therefore, one method often employed in processing emails is to “normalize’ these values’, so that all URLs are treated the same, all numbers are treated the same, etc. For example, we could replace each URL in the email with the unique string “httpaddr” to indicate that a URL was present. This has the effect of letting the spam classifier make a classification decision based on whether any URL was present, rather than whether a specific URL was present. This typically improves the performance of a spam classifier, since spammers often randomize the URLs, and thus the odds of seeing any particular URL again in a new piece of spam is very small.

We have already implemented the following email preprocessing steps: lower-casing; removal of HTML tags; normalization of URLs, email addresses, and numbers. In addition, words are reduced to their stemmed form. For example, “discount”, “discounts”, “discounted” and “discounting” are all replaced with “discount”. Finally, we removed all non-words and punctuation. The result of these preprocessing steps is shown in lower image of Figure 1.

Experiments

- This project will involve your implementing classification algorithms. Before you can build these models and measures their performance, split your training data (i.e. spam train.txt) into a training and validate set, putting the last 1000 emails into the validation set. Thus, you will have a new training set with 4000 emails and a validation set with 1000 emails. **Explain why measuring the performance of your final classifier would be problematic had you not created this validation set.**

Ans: In a classification work flow, training data set are in two categories (training and validation). Then the test data set is for testing. The problem with not creating a validation set can cause the performance measure to perform poorly due to inaccurate prediction.

- Transform all of the data into **feature vectors**. Build a vocabulary list using only the 4000 email training set by finding all words that occur across the training set. Note that we assume that the data in the validation and testsets is completely unseen when we train our model, and thus we do not use any information contained in them. Ignore all words that appear in fewer than X = 30 emails of the 4000 email training set. This is both a means of preventing overfitting and of improving scalability. For each email, transform it into a feature vector x where the ith entry, xi, is 1 if the ith word in the vocabulary occurs in the email, and 0 otherwise.
- Train the linear classifier such as Naive Bayes using your training set. How many mistakes are made before the algorithm terminates? Next, classify the emails in your validation set. What is the validation error? Explain your results.

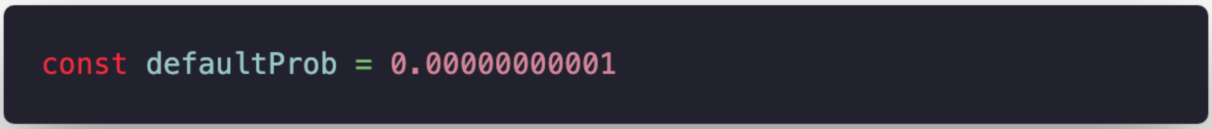
Ans: The algorithm makes about 1.575% mistake during the training phase. In addition, validation error is about 2.7% while testing the validation data set.

- Explore some other algorithms to solve spam filter problem. And demonstrate your thoughts.

Design and Implementation

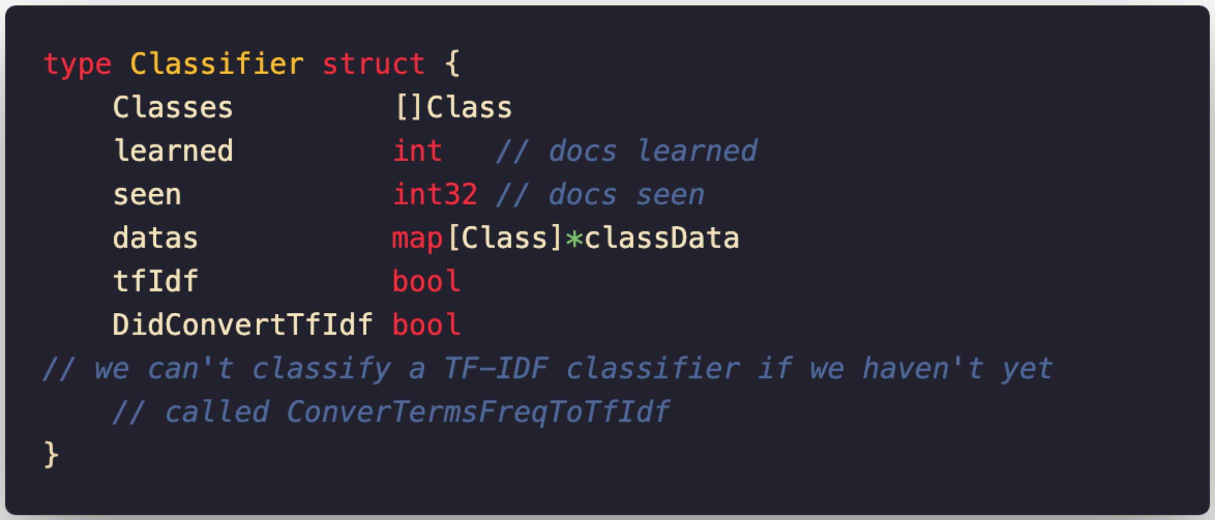
Implementation of Bayesian

defaultProb is the tiny non-zero probability that a word we have not seen before appears in the class.



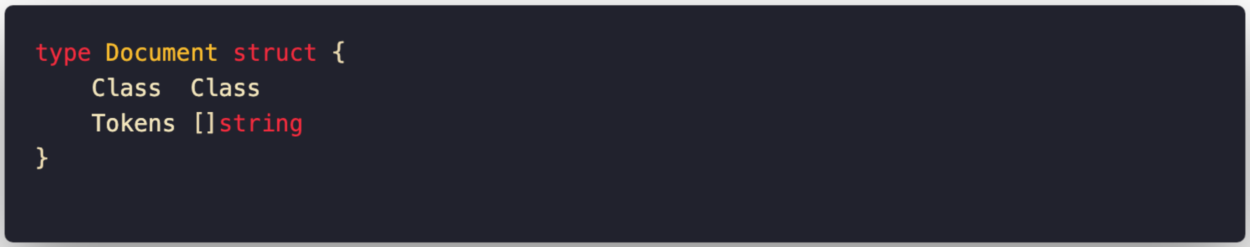
```go
const defaultProb = 0.00000000001
```

Classifier implements the Naive Bayesian Classifier



```go
type Classifier struct {
	Classes         []Class
	learned         int   // docs learned
	seen            int32 // docs seen
	datas           map[Class]*classData
	tfIdf           bool
	DidConvertTfIdf bool
// we can't classify a TF-IDF classifier if we haven't yet
	// called ConverTermsFreqToTfIdf
}
```

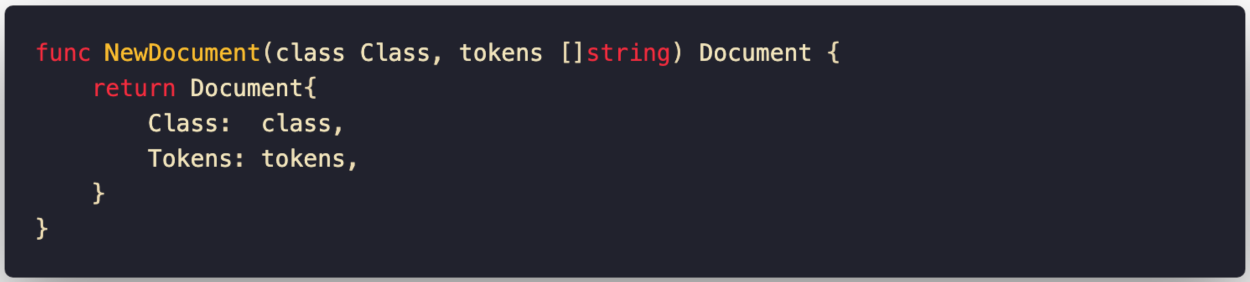
Document is a group of tokens with certain class



```go
type Document struct {
	Class  Class
	Tokens []string
}
```

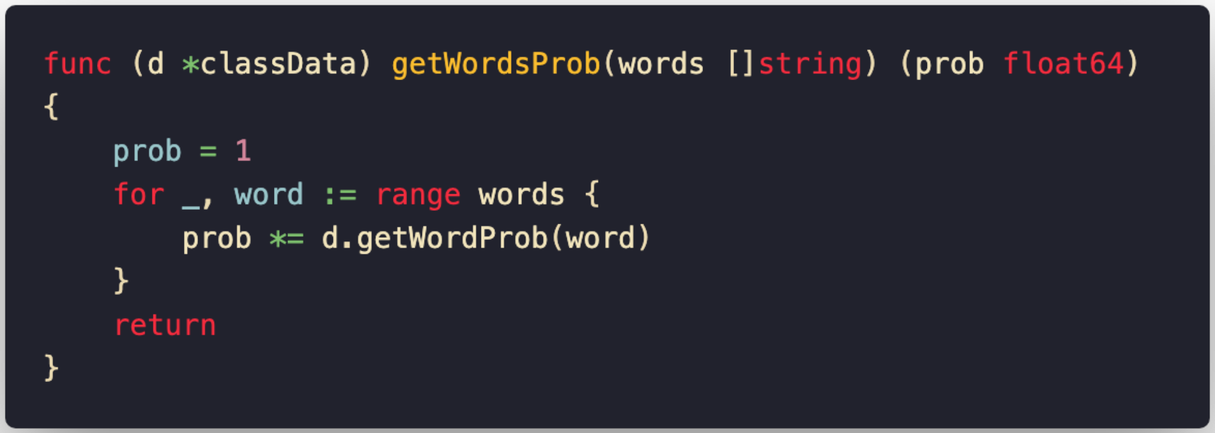
NewDocument return new Document

getWordProb returns P(W | C_j) -- the probability of seeing a particular word W



```
func NewDocument(class Class, tokens []string) Document {
    return Document{
        Class:  class,
        Tokens: tokens,
    }
}
```

in a document of this class. *getWordsProb* returns P(D|C_j)
-- the probability of seeing this set of words in a document of this class. Note that words should not be empty, and this method of calculation is prone to underflow if there are many words and their individual probabilities are small.



```
func (d *classData) getWordsProb(words []string) (prob float64)
{
    prob = 1
    for _, word := range words {
        prob *= d.getWordProb(word)
    }
    return
}
```

The term frequency - inverse document frequency (Tf-idf) is another alternative for characterizing text documents. It can be understood as a weighted term frequency, which is especially useful if stop words have not been removed from *the text corpus. The Tf-idf approach assumes that the importance of a word is inversely proportional to how often it occurs across all documents. Although Tf-idf is most commonly used to rank documents by relevance in different text mining tasks, such as page ranking by search engines, it can also be applied to text classification via naive Bayes.*

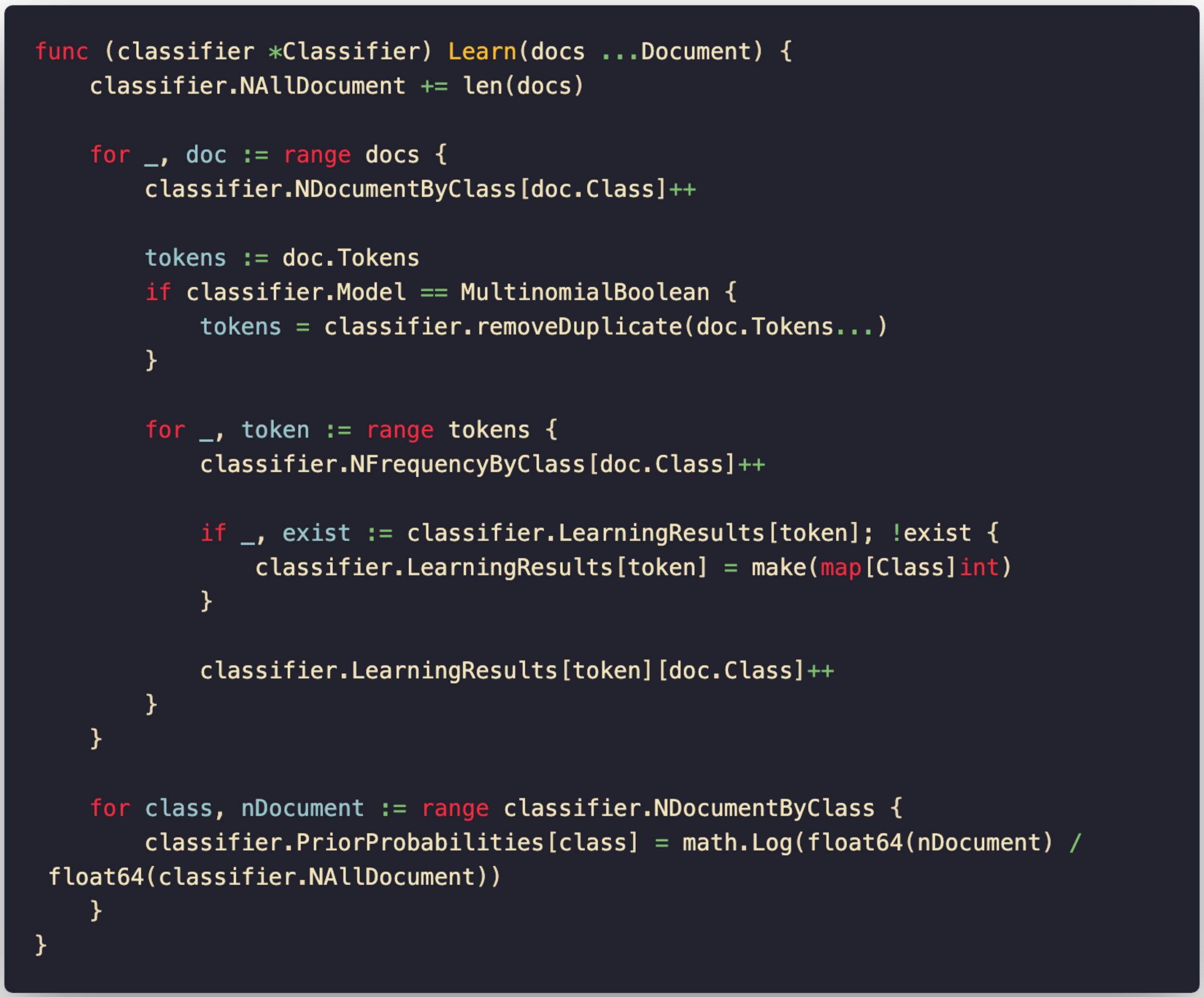
```go
type Classifier struct {
    Model              Model
    LearningResults    map[string]map[Class]int
    PriorProbabilities map[Class]float64
    NDocumentByClass   map[Class]int
    NFrequencyByClass  map[Class]int
    NAllDocument       int
}
```

```go
func NewClassifierFromFile(path string) (Classifier, error) {
    classifier := Classifier{}

    fl, err := os.Open(path)
    if err != nil {
        return classifier, err
    }
    defer fl.Close()

    err = gob.NewDecoder(fl).Decode(&classifier)
    if err != nil {
        return classifier, err
    }

    return classifier, err
}
```

NewClassifier returns a new classifier. The classes provided should be at least 2 in number and unique, or this method will panic

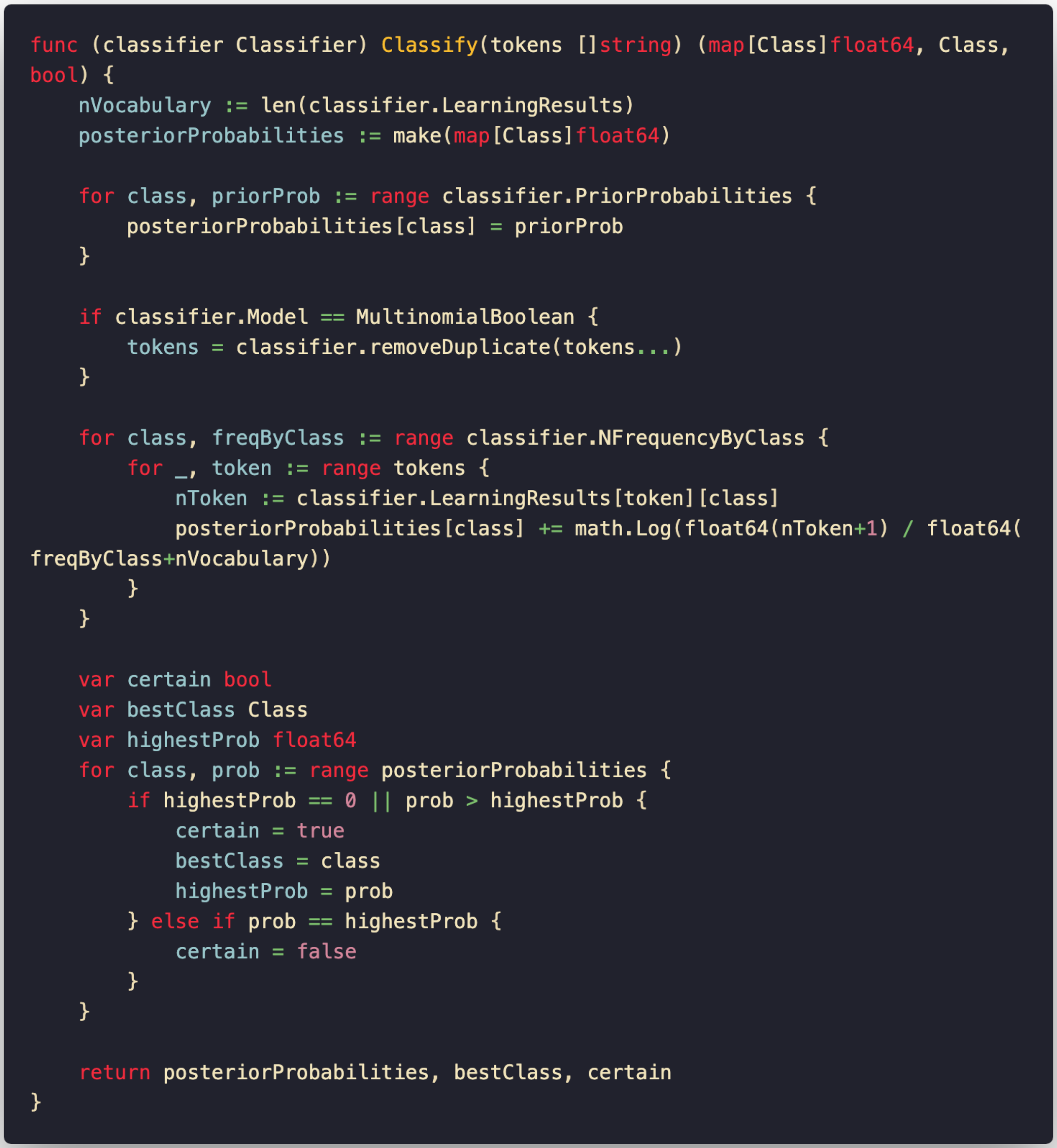
```go
func NewClassifier(model Model) Classifier {
    return Classifier{
        Model:              model,
        LearningResults:    make(map[string]map[Class]int),
        PriorProbabilities: make(map[Class]float64),
        NDocumentByClass:   make(map[Class]int),
        NFrequencyByClass:  make(map[Class]int),
    }
}
```

Learn will accept new training documents for supervised learning.



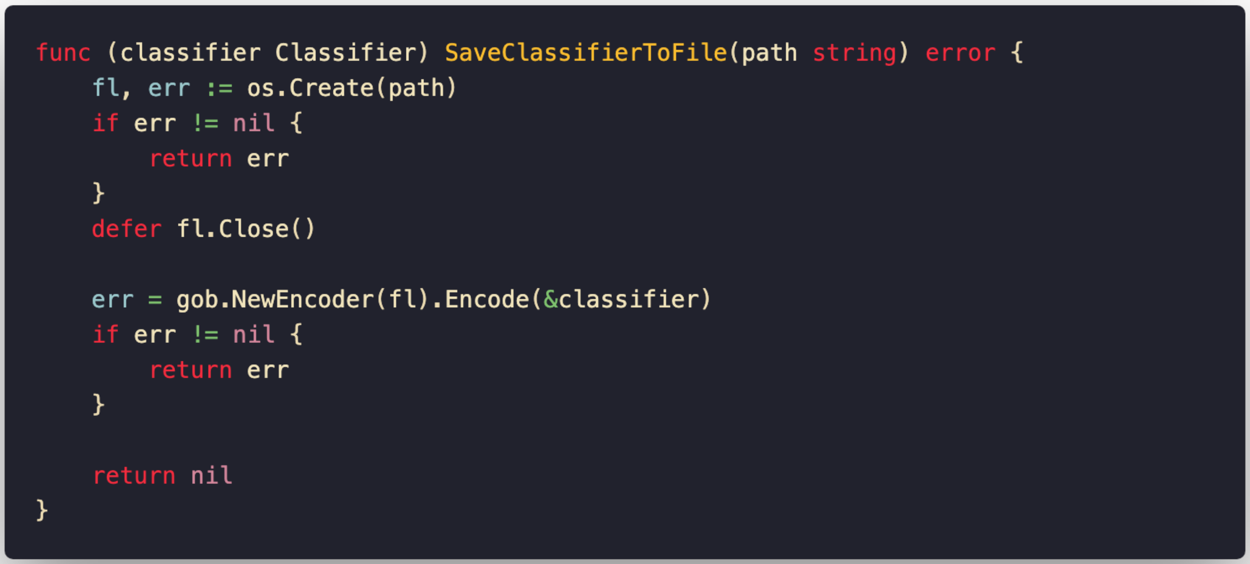
```go
func (classifier *Classifier) Learn(docs ...Document) {
    classifier.NAllDocument += len(docs)

    for _, doc := range docs {
        classifier.NDocumentByClass[doc.Class]++

        tokens := doc.Tokens
        if classifier.Model == MultinomialBoolean {
            tokens = classifier.removeDuplicate(doc.Tokens...)
        }

        for _, token := range tokens {
            classifier.NFrequencyByClass[doc.Class]++

            if _, exist := classifier.LearningResults[token]; !exist {
                classifier.LearningResults[token] = make(map[Class]int)
            }

            classifier.LearningResults[token][doc.Class]++
        }
    }

    for class, nDocument := range classifier.NDocumentByClass {
        classifier.PriorProbabilities[class] = math.Log(float64(nDocument) /
 float64(classifier.NAllDocument))
    }
}
```

LogScores produces "log-likelihood"-like scores that can be used to classify documents into classes. The value of the score is proportional to the likelihood, as determined by the classifier, that the given document belongs to the given class. This is true even when scores returned are negative, which they will be (since we are taking logs of probabilities). The index j of the score corresponds to the class given by c.Classes[j]. Additionally returned are "inx" and "strict" values. The inx corresponds to the maximum score in the array. If more than one of the scores holds the maximum values, then strict is false. Unlike c.Probabilities(), this function is not prone to floating point underflow and is relatively safe to use.



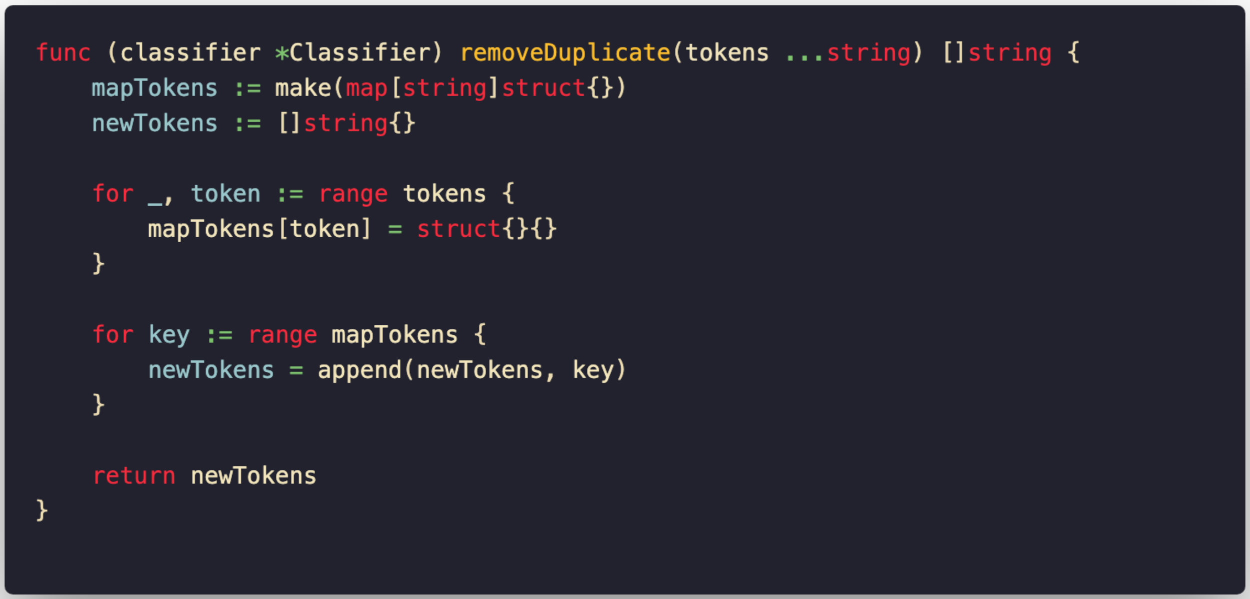
```
func (classifier Classifier) Classify(tokens []string) (map[Class]float64, Class,
bool) {
    nVocabulary := len(classifier.LearningResults)
    posteriorProbabilities := make(map[Class]float64)

    for class, priorProb := range classifier.PriorProbabilities {
        posteriorProbabilities[class] = priorProb
    }

    if classifier.Model == MultinomialBoolean {
        tokens = classifier.removeDuplicate(tokens...)
    }

    for class, freqByClass := range classifier.NFrequencyByClass {
        for _, token := range tokens {
            nToken := classifier.LearningResults[token][class]
            posteriorProbabilities[class] += math.Log(float64(nToken+1) / float64(
freqByClass+nVocabulary))
        }
    }

    var certain bool
    var bestClass Class
    var highestProb float64
    for class, prob := range posteriorProbabilities {
        if highestProb == 0 || prob > highestProb {
            certain = true
            bestClass = class
            highestProb = prob
        } else if prob == highestProb {
            certain = false
        }
    }

    return posteriorProbabilities, bestClass, certain
}
```

Saves machine learning model to a file.



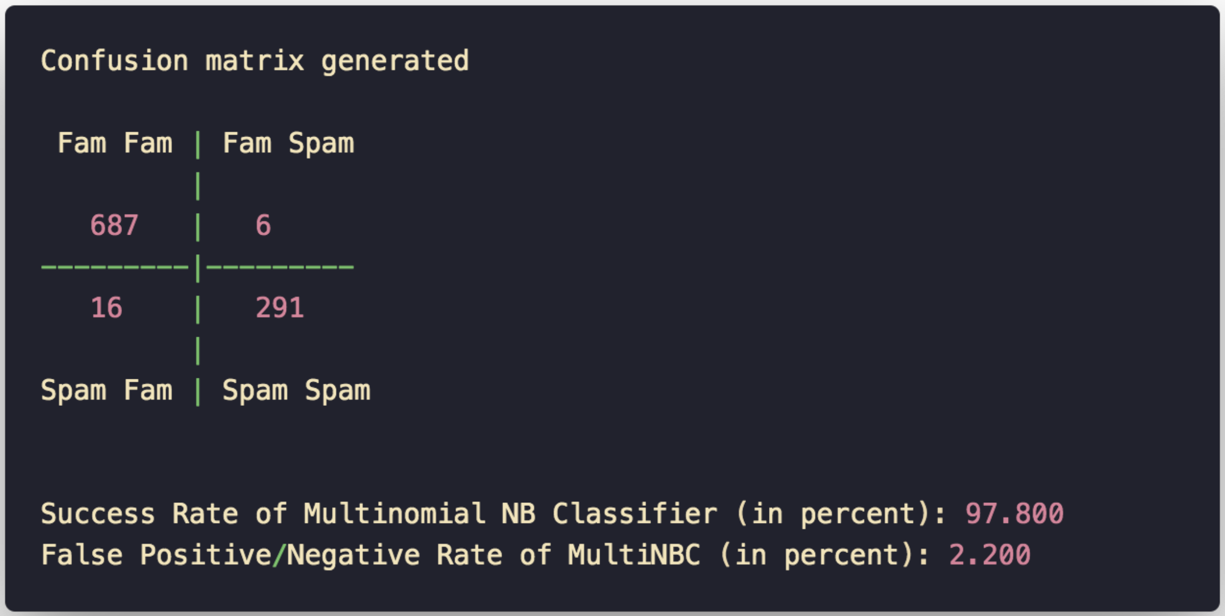
```go
func (classifier Classifier) SaveClassifierToFile(path string) error {
    fl, err := os.Create(path)
    if err != nil {
        return err
    }
    defer fl.Close()

    err = gob.NewEncoder(fl).Encode(&classifier)
    if err != nil {
        return err
    }

    return nil
}
```

removeDuplicate Removes duplicate tokens from file.



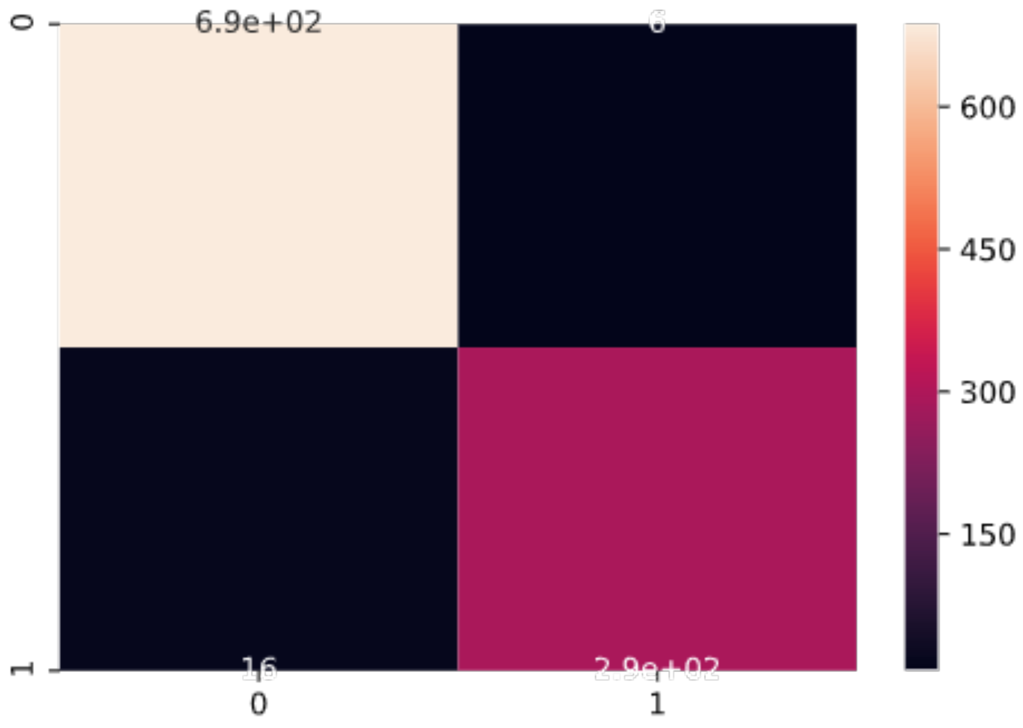
```go
func (classifier *Classifier) removeDuplicate(tokens ...string) []string {
    mapTokens := make(map[string]struct{})
    newTokens := []string{}

    for _, token := range tokens {
        mapTokens[token] = struct{}{}
    }

    for key := range mapTokens {
        newTokens = append(newTokens, key)
    }

    return newTokens
}
```

Result Analysis



```
Confusion matrix generated

 Fam Fam | Fam Spam
         |
   687   |   6
---------|---------
   16    |   291
         |
Spam Fam | Spam Spam


Success Rate of Multinomial NB Classifier (in percent): 97.800
False Positive/Negative Rate of MultiNBC (in percent): 2.200
```

For benchmarking our classifier, we used confusion matrix to represent false positives and False negatives.



This confusion matrix represents the true positive and true negative rates with heatmap. See the code at https://github.com/diabloxenon/Spamaway.git

Conclusion

Opinions from online digital media are increasingly used by individuals and organizations for making purchase decisions and marketing and product design. Positive opinions often mean profits and fames for businesses and individuals. This is a strong incentives for people to game the system and manipulate user sentiment by posting fake opinions or reviews to promote or to discredit some target products. Such individuals are called opinion spammers and their activities are called opinion spamming.

It is very unusual to having 100% success from a model. Obviously, it is due to small training and test dataset. I have tested my own emails using the model. It turned out that it is not as effective as my existing paid spam filter. It makes sense. There are many ways we can improve the model. If the model trains with sufficient data, it will deliver more accurate results.

References

1. Sebastian Raschka, Naive Bayes and Text Classification https://arxiv.org/pdf/1410.5329v3.pdf

2. Naive Bayes Spam Filtering https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering

3. Spam Dataset https://www.kaggle.com/veleon/ham-and-spam-dataset#0007.859c901719011d56f8b652ea071c1f8b

4. Naive Bayes Classifier https://en.wikipedia.org/wiki/Naive_Bayes_classifier

5. Naman Bishnoi's Spamaway project https://github.com/diabloxenon/Spamaway.git