https://dx.doi.org/10.22161/ijaers.612.64 ISSN: 2349-6495(P) | 2456-1908(O)

Bayesian Spam Filter for Wolaytta

Tewodros Abebe

Computer Science, Wolaita Sodo University, Ethiopia
teddy4students@gmail.com

Abstract**—** *The increasing volume of spam has become a serious threat not only to the Internet, but also to society. This paper presents a C++ code to implement a spam filtering based on the use of Bayesian spam filtering for Wolaytta text. Our experiments indicate that using this approach to filter spam e-mails is a good approach for languages like Wolaytta.*

Keywords *—* ***Spam filter, Naïve Bayesian classification, Wolaytta.***

I. INTRODUCTION

With the advancement in electronic and computer technology there is an explosive growth in the use of computers for processing information. In today's computer-connected society, among the existing forms of communication email has become the fastest and most economical form of available communication.

According to recent surveys, 60% of all e-mail traffic is spam [1]. In this way a great amount of bandwidth is wasted and the e-mail systems are overloaded. Due to the above serious problems, measures must be taken to deal with the spam phenomenon. The best such measure has proven to be spam filtering.

This paper demonstrates the use of Bayes' formula in filtering spam messages of Wolaytta text. Achieving the goal for local languages like Wolaytta would also provide novel knowledge about cognition, understanding, and mechanism of spam filtering with the possibility of considering some of the features of the language.

II. BAYESIAN PROBABILITY

Bayesian probability is one of the major theoretical and practical frameworks for reasoning and decision making under uncertainty [2]. The historical roots of this theory lie in the late 18th, early 19th century, with Thomas Bayes [2] and Pierre-Simon de Laplace [3]. It was "forgotten" for a long time, and began to be re-appreciated in different application domains, during various periods of the 20th century. Hence, Bayesian probability was never developed as one single, homogeneous piece of scientific activity. Bayesian concepts, methods and solutions for different applications became known for decades under various names: the Bayesian approach to uncertainty reasoning, Bayesianism, the Bayesian framework, the Bayesian paradigm, plausible inference, and Bayesian reasoning

A. Bayes' Formula

In basic terms, Bayes' Formula allows us to determine the probability of an event occurring, based on the probabilities of two or more independent evidentiary events [4]. Mathematically, the general formula is represented as: **P(Ej|F)**

$$P(Ej|F) = \frac{P(F|Ej)P(Ej)}{\sum P(F|Ei)P(Ei)}$$

Assuming that the variables a and b are the probabilities of two evidentiary events, the probability would be equal to:

$$\frac{ab}{ab + (1-a)(1-b)}$$

For three evidentiary events a, b, and c, the formula expands so the probability is equal to:

$$\frac{abc}{abc + (1-a)(1-b)(1-c)}$$

In this fashion, the formula can be expanded to accommodate any number of evidentiary events. The Bayes formula lets us combine the probability of multiple independent events into one number with a range of 0.0 to 1.0. Here the Bayes' formula is used to figure out the probability that a message is spam based on the words appearing in it.

The block diagram below in figure 1 shows how the spam filter module is designed. The goal of the filter program is to filter spam messages as it occurs as a legitimate mail message. The program takes data to train the train_data_table, a table which is used to store words (tokens) that appear in spam and non-spam messages along with their frequency both in spam and non-spam messages. It also has columns that hold calculated values of probability and spamicity. The filter:

I. Parses each word in sample spam and legitimate (non-spam) messages. These messages are used to train the filter with possible tokens that appear in spam message.

https://dx.doi.org/10.22161/ijaers.612.64 *ISSN: 2349-6495(P) | 2456-1908(O)*

II. For each token it uses its frequency both in spam and legitimate messages. The frequency is then used to calculate the probability of each token being as it being appears in spam and legitimate messages.
III. The calculated probability then yields the spamicity.

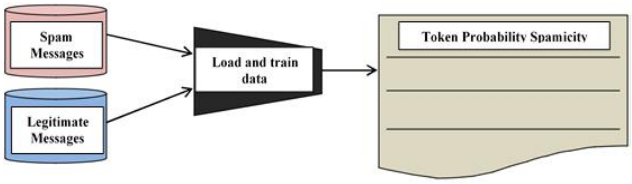


Fig. 1: Spam Filter Module

The first thing we need to do is to provide the filter with a couple of spam and legitimate messages. This will allow the filter to train itself the difference between spam and non-spam messages. When we humans accidentally read a spam message, we almost immediately recognize it as spam because of certain key words (such as "viagra" and "mortgage") or phrases (such as "Get your free porn here!"). Instinctively, we know that a message containing these words or phrases is spam because of our experience in dealing with junk mail. The opposite is true as well. We can almost instantly look at a message from our mother (containing phrases such as "When are you going to get married so I can have grandchildren?") or our boss (containing phrases such as "Less computer solitaire and more work if you want a paycheck this Friday") and know they're not spam.

III. WOLAYTTA LANGUAGE

Wolaytta is an Omotic language family which is a branch of the Afro-Asiatic language phylum spoken in the Wolaytta Zone and some parts of the Southern Nations, Nationalities, and People's Region of Ethiopia [6]. The term 'Wolaytta' is used as the name by which the people refer to themselves, their region and their language. The Wolaytta language is written in the Latin alphabet. It is one of the languages known as it has complex morphology.

IV. THE SYSTEM (BAYESIAN FILTER)

The system developed here implements the Bayes' filter. It is named 'Wolaytta_Spam_Filter'. To develop the system, I used C++ programming language that I am familiar with. The system uses a single table (vector) to store tokens that are supposed to be both in spam and non-spam messages. The messages are written in Wolaytta.

A. Training Data

The more we train our filter the more it will become accurate. We "train" the filter by showing it a bunch of mail messages, and telling it whether the message is spam. Whenever we show a message to the filter, it finds every word in the message and stores it (along with how many times it occurred) in a database.

To train the system I translated the most familiar spam messages occur in English to Wolaytta text and give the system the text in .txt file format. This due to the problem of inaccessibility of any written text for this language in Examples of the spam and non-spam messages used for the training 'Wolaytta_Spam_Filter' are listed below.

Table I: Sample Spam and Non-Spam messages of Wolaytta

Spam and Non-Spam messages
Spam Messages
1. Tanni kehini nena sikio gishawu tannara gayitanawu koiko hachi tawu silikia shocha. Ta nena dossiyoo gishsha ta neyoo kittido ta pootuwan beada ne tana dossiko tayoo neeni silkkiyaan tani neeko yaada nenaara gayttana.
2. Dvyaa Lotoriyaa gakanadani koiko tayo nenni birra bankirra yedana mala tomooseyis. Hega otanawu ne koiko tanni niyo esuwara polissada nena America bitta yedana. Nena dvyaa qaaday gakkido gishaw, ne bankkiyaa payduwaa nussi issi agnnaa gakkanaashin eesuwan xaafa. Ne hegaa xaafa simishiin nu neyoo nena ekkida giya woraqataa yedana.
Non Spam Messages
1. Tani issi shanne uddppu xeetane eudpu tamanu marotetaa layttan yeletasi. Hega gioge ta layitai hae eshattamapee aleissi gioga. La Taddalla aymala deay waanada xayadii? Ne ossoy aymalee? Ne naati ubbayka saroo? Ne siiqo machchiyaa aymalee? Taani neeko mata wodiyaan yaana. Hegee gakkanaashin saro dea.
2. Ha gidooni intte ooso keettay oosanchchasi oorata oosuwan qadaa kessidogaa kessidoogaa siyaasi shiinawude gakkanawu takii? Ubbankka neesi wontto silkkiaa shoccana. Saro taka

For training purpose, the filter takes tokens both form spam and non-spam messages. For this project I used 4 spam and 4 non-spam messages for training purpose. The messages listed above only show the content of the messages.

Here I used Wireshark, a network analysis tool, for capturing the actual messages as if it is sent through E-Mail. While sending the E-Mail Wireshark uses http protocol and hence, I extracted the content of the messages from the raw packets. During spam filtering the main focus is on the actual message content so that other parts of the

https://dx.doi.org/10.22161/ijaers.612.64 *ISSN: 2349-6495(P) | 2456-1908(O)*

message such as Header parts are not important to be listed here in training data. As a result, the header and body fields along with their values are not included in the message texts listed above.

All title and author details must be in single-column format and must be centered.

The messages are hence used to populate the filter table. The probabilities are calculated using an easy probability formula used to compute the probability value. In order to calculate the probability of each token the filter use the frequency of the token as it appears on spam and non-spam messages. The values of the calculated probabilities are used to compute the spamicity of a given token.

This probability value assigned to each word is commonly referred to as spamicity, and ranges from 0.0 to 1.0. If spamicity value for a given word is greater than 0.5 then the message containing the word/token is likely to be spam, while a spamicity value less than 0.5 indicates that a message containing the word is likely to be ham (non-spam). A spamicity value of 0.5 is neutral, meaning that it has no effect on the decision as to whether a message is spam or not.

Some spam filtering applications use a separate table for spams and non-spam messages. Here I used a common table for both spam and non-spam messages. This approach is easy for reading the values that correspond to a particular token. A count of messages used for training is also kept on the database to calculate the probabilities.

Table II: Sample Spam and Non-Spam messages of Wolaytta

Token	Spam frequency	Non-spam frequency	Pr(spam)	Pr(non-spam)	Spamicity
Tanni	4	0	1	0	1
Kehini	1	0	0.25	0	1
Nena	3	0	0.75	0	1
Aleissi	0	1	0	0.25	0
Hega	1	1	0.25	0.25	0.5
Nyio	2	0	0.5	0	1

The implementation of the filter to create the above train database uses the following structures:

```
typedef struct token_table
{
        string word;
        int id;
        int non_spam_freq;
        int spam_freq;
        float ps;
        float pns;
        float spamicity;
}token_table;
```

The "token_table" is a structure used to hold tokens parsed for spam and non-spam messages. Non-spam and spam fields store integer values that tell the number of appearances of a word in spam and non-spam messages. These values are used for finding the probabilities of each word that exists within the messages.

Once the Bayesian filter has the list of tokens in the table, it searches the spam and non-spam tokens databases for these tokens. These databases of tokens are created and updated whenever the Bayesian filter is "trained" on a new message. This implies training only requires messages that are supposed to be spam and non-spam.

If a token from the message is found in the databases, the Bayesian filter calculates the token's spamicity based on the following variables:

I. The frequency of the token in spam messages that the filter has been trained on
II. The frequency of the token in ham messages that the filter has been trained on
III. The number of spam messages the filter has been trained on
IV. The number of ham messages the filter has been trained on

The algorithm used to calculate a token's spamicity from these pieces of information is as follows:

$$\text{Ham probability} = \frac{\text{Token frequency in ham messages}}{\text{Number of ham messages trained on}}$$

Equation 1

https://dx.doi.org/10.22161/ijaers.612.64 *ISSN: 2349-6495(P) | 2456-1908(O)*

$$\text{Ham probability} = \frac{\text{Token frequency in ham messages}}{\text{Number of ham messages trained on}}$$

Equation 2

If either Ham probability or Spam probability are greater than 1.0, it sets them equal to 1.0.

$$\text{Spamicity} = \frac{\text{Spam probability}}{\text{Ham probability} + \text{Spam probability}}$$

Equation 3

The structure used to store the tokens along with their structure members are shown below:

```
typedef struct train_Data

{

        vector<token_table> trainTable;

        int non_spam_word;

        int spam_word;

}train_Data;
```

The fields 'spam_word' and 'non_spam_word' are integer variables used to store the number of spam and non-spam messages used to train the train database. The vector holds tokens along with the fields specified in 'token_table'.

First messages in non-spam category are loaded and then messages in spam category. These processes are handled by function:

loadTrainDataSpam()

loadTrainDataHam()

after the tokens in spam and non-spam messages are parsed and populated along with their frequency in spam and non-spam messages. The next step is calculating the probability and spamicity of each token. The function which calculates these values is:

calculateProbability()

This function takes the values populated by the previous function, *loadTrainDataSpam()* and *loadTrainDataHam()*.

These three functions use the formulas listed above, Equation1, Equation2 and Equation3.

Loading training messages involves adding the parsed tokens in to the train_table. For such functions I used the function:

add_token_ham(token)

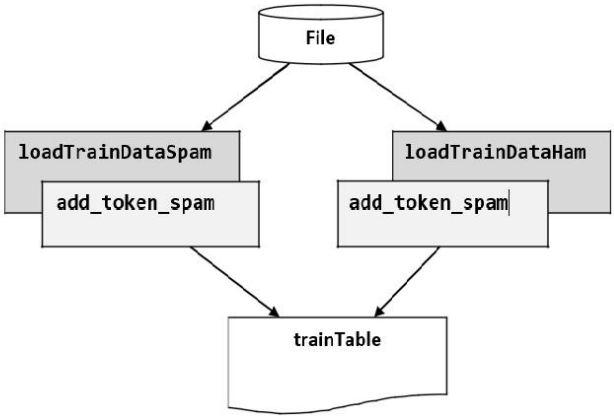


Fig.2: Filter processing

Next, I'm gone to let the filter try to decide if a message is spam or not, based on what we've told it about what spam looks like and how it differs from non-spam. Let's pretend a mail message is sent to the filter, which looks like this:

"Tanni kehini nena sikio gayitanawu koiko hachi tawu silikia shocha"

The filter scans through the message, creating a list of every word it knows about (in other words, every word in the message that's also in the token databases). Once the filter has the list of words it knows about, for each word it calculates the probability that the word appears in spam based on the frequency data in the token databases.

V. Testing and Evaluation of the New System

This section provides the performance assessment of the spam filter followed by testing the adequacy of the system.

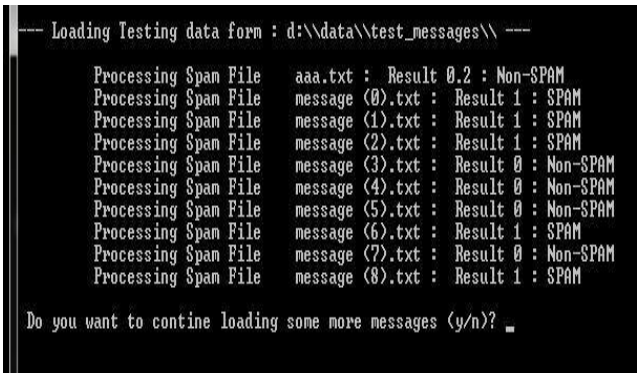
The filter takes as input a directory with email messages (both spam and non-spam) for testing the filter specified by the user. In order to use the program and test some messages a user requires:

I. A directory\folder with spam messages used to train the filter
II. A directory\folder with non-spam messages used to train the filter
III. A directory\folder with messages (both spam and non-spam) for testing the filter

The program continues well and shows all train database and the result for the tested messages. It filters the message as spam and non-spam and shows the result as follows:

Fig 3: Processing Spam File

https://dx.doi.org/10.22161/ijaers.612.64 *ISSN: 2349-6495(P) | 2456-1908(O)*



Most of developers of this system train and evaluate their system with thousands of messages. But for this case due to unavailability of Wolaytta text, I translate some messages and try to evaluate on the texts that I trained the system and with some more. To evaluate the filter's performance, I performed the following test case:

The inbox contains 10 email messages: 5 spams and 5 non-spams.

- Total number of email messages: 10
- Total number of non-spam messages: 5
- Total number of spam messages: 5
- Total number of email messages classified as non-spam: 5
- Total number of email messages classified as spam: 5
- Total number of non-spam messages classified as spam messages: 0
- Total number of spam messages classified as non-spam messages: 0
- Accuracy: 100%

This doesn't show precisely the accuracy of the system as if it is trained with small number of training sets and small number of testing sets of data.

VI. CONCLUSION

As we have experienced, most of the email messages that are sent for us daily are spam. According to different researchers among the existing different techniques used for text classification, Bayesian analysis filters spam with high accuracy.

Here I tried to develop a Bayesian spam filter using C++ programming language. I read different articles to understand the Bayesian filtering technique and I developed a system that is capable to do filtering messages as spam and non-spam. What we need to do on the system is to train once and we are done. After training the filter, it becomes capable of filtering spam with high accuracy as shown in the evaluation section.

REFERENCES

[1] Fetterly, Dennis, Mark Manasse, and Marc Najork. "Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages." *Proceedings of the 7th International Workshop on the Web and Databases: colocated with ACM SIGMOD/PODS 2004*. ACM, 2004.

[2] Bayes, T. (1958). Essay towards solving a problem in the doctrine of chances. *Biometrika*, *45*, 293-315.

[3] Bruyninckx, H. (2002). Bayesian probability. *CiteSeer, não publicado em con*, 81.

[4] Metsis, V., Androutsopoulos, I., & Paliouras, G. (2006, July). Spam filtering with naive bayes-which naive bayes?. In *CEAS* (Vol. 17, pp. 28-69).

[5] Lamberti, M., & Sottile, R. (1997). *The Wolaytta Language* (pp. 665-665). Rüdiger Köppe Verlag.*Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.

[6] Wakasa, M. (2014). *A sketch grammar of Wolaytta*. Japan Association for Nilo-Ethiopian Studies.