Toward Time-Sensitive Structure Analysis for SPAM Filtering:

A Data Mining Approach

(English reproduction)

Joshua Olson and Atsushi Inoue

# Purpose

This paper seeks to reproduce the paper Dr. Kobayashi et al. by using decision trees and discrete time periods as an approach to SPAM filtering.

# Approach

## Building the Thesauri

First, we determined that we should recreate the four periods of the original paper, shortening the length of time, but preserving the quantity of SPAM and HAM messages processed. Only the first period contains few messages due to the start of the project occurring on March 10th.

|  |  |  |
| --- | --- | --- |
| Period | SPAM | HAM |
| March 1 - 15 | 172 | 187 |
| March 16 - 31 | 659 | 389 |
| April 1 - 15 | 552 | 400 |
| April 16 - 30 | 456 | 388 |
| May 1 - 15 | 546 | 365 |

We then created a python script to scan all emails in the period and built a dictionary of word frequencies for SPAM and HAM using python’s natural language toolkit (NLTK) to clean html tags and tokenize the message bodies.

Here we extracted the top 50% of the popular SPAM words from the frequency list. Then we compared the SPAM frequency list with the HAM frequency list and extracted the SPAM words with a frequency of one that did not appear in the HAM frequency list.

## Building the dataset

At this point we examined every SPAM and HAM message extracting the following items using an SMTP parser and regular expressions. All reasonable attempts were made to follow the original paper.

### IP Address

We extracted all IP Addresses in the received field then hashed the value using the MD5 hashing algorithm and a hex digest form to give a compact representation of this data.

### Matching Degree of domain names between Message-ID and Sender fields

This was done by extracting the domain names from the Message-ID and From fields using regular expressions then computing the Levenshtein distance to determine the degree of matching

### Subject

The subject was extracted and hashed the same as the IP addresses.

### Name

This was extracted from the from field and hashed also.

### Content Type

1 – was used for messages that contained HTML only in the message body.

2 – was used for messages that contained text only in the message body.

3 – was used for messages that contained both text and HTML versions of the message body.

### Attachments

1 – was used for messages with no attachments.

2 – was used for messages with text attachments.

3 – was used for messages with non-text attachments.

4 – was used for messages with mixed attachments.

### Number of URLs in the message content

This is done with a match all URL regular expression on the body.

### URL ratio in the message

This is the percent of URL character divided by the total number of character in the message body.

### SPAM word ratio (SPAM %)

Number of words that appear in the list of popular SPAM words or the list of definite SPAM words divided by the total number of words in the message body.

### SPAM Degree

This is calculated as described in equations 1, 2 & 3 in the original paper with w1 and w2 from the original paper.

### Class Label

## Cross validation

At this point we examined every SPAM and HAM message and have extracted the raw data to be examined. Using python we divided each period data into three sets using a uniform pseudo-random number generator. From here we combined we used two-thirds of the data as a training file and the rest as the testing file for Weka. We then ran all three combinations through Weka for cross validation.

# Results