NAME

bogofilter - fast Bayesian spam filter

SYNOPSIS

bogofilter [help options | classification options | registration options | parameter options | info options] [general options] [config file options]

where

help options are:

[-h] [--help] [-V] [-Q]

classification options are:

[-p] [-e] [-t] [-T] [-u] [-H] [-M] [-b] [-B *object* ...] [-R] [general options] [parameter options] [config file options]

registration options are:

[-s | -n] [-S | -N] [general options]

general options are:

[-c *filename*] [-C] [-d *dir*] [-k *cachesize*] [-l] [-L *tag*] [-I *filename*] [-O *filename*]

parameter options are:

[-E *value[,value]*] [-m *value[,value][,value]*] [-o *value[,value]*]

info options are:

[-v] [-y *date*] [-D] [-x *flags*]

config file options are:

[--*option=value*]

Note: Use **bogofilter --help** to display the complete list of options.

DESCRIPTION

Bogofilter is a Bayesian spam filter. In its normal mode of operation, it takes an email message or other text on standard input, does a statistical check against lists of "good" and "bad" words, and returns a status code indicating whether or not the message is spam. Bogofilter is designed with a fast algorithm, uses the Berkeley DB for fast startup and lookups, coded directly in C, and tuned for speed, so it can be used for production by sites that process a lot of mail.

THEORY OF OPERATION

Bogofilter treats its input as a bag of tokens. Each token is checked against a wordlist, which maintains counts of the numbers of times it has occurred in non-spam and spam mails. These numbers are used to compute an estimate of the probability that a message in which the token occurs is spam. Those are combined to indicate whether the message is spam or ham.

While this method sounds crude compared to the more usual pattern-matching approach, it turns out to be extremely effective. Paul Graham's paper **A Plan For Spam**[1] is recommended reading.

This program substantially improves on Paul's proposal by doing smarter lexical analysis. Bogofilter does proper MIME decoding and a reasonable HTML parsing. Special kinds of tokens like hostnames and IP addresses are retained as recognition features rather than broken up. Various kinds of MTA cruft such as dates and message-IDs are ignored so as not to bloat the wordlist. Tokens found in various header fields are marked appropriately.

Another improvement is that this program offers Gary Robinson's suggested modifications to the calculations (see the parameters robx and robs below). These modifications are described in Robinson's paper **Spam Detection**[2].

Since then, Robinson (see his Linux Journal article **A Statistical Approach to the Spam Problem**[3]) and others have realized that the calculation can be further optimized using Fisher's method. **Another improvement**[4] compensates for token redundancy by applying separate effective size factors (ESF) to spam and nonspam probability calculations.

In short, this is how it works: The estimates for the spam probabilities of the individual tokens are combined using the "inverse chi-square function". Its value indicates how badly the null hypothesis that the message is just a random collection of independent words with probabilities given by our previous estimates fails. This function is very sensitive to small probabilities (hammish words), but not to high probabilities (spammish words); so the value only indicates strong hammish signs in a message. Now using inverse probabilities for the tokens, the same computation is done again, giving an indicator that a message looks strongly spammish. Finally, those two indicators are subtracted (and scaled into a 0-1-interval). This combined indicator (bogosity) is close to 0 if the signs for a hammish message are stronger than for a spammish message and close to 1 if the situation is the other way round. If signs for

both are equally strong, the value will be near 0.5. Since those message don't give a clear indication there is a tristate mode in bogofilter to mark those messages as unsure, while the clear messages are marked as spam or ham, respectively. In two-state mode, every message is marked as either spam or ham.

Various parameters influence these calculations, the most important are:

robx: the score given to a token which has not seen before. robx is the probability that the token is spammish.

robs: a weight on robx which moves the probability of a little seen token towards robx.

min-dev: a minimum distance from .5 for tokens to use in the calculation. Only tokens farther away from 0.5 than this value are used.

spam-cutoff: messages with scores greater than or equal to will be marked as spam.

ham-cutoff: If zero or spam-cutoff, all messages with values strictly below spam-cutoff are marked as ham, all others as spam (two-state). Else values less than or equal to ham-cutoff are marked as ham, messages with values strictly between ham-cutoff and spam-cutoff are marked as unsure; the rest as spam (tristate)

sp-esf: the effective size factor (ESF) for spam.

ns-esf: the ESF for nonspam. These ESF values default to 1.0, which is the same as not using ESF in the calculation. Values suitable to a user's email population can be determined with the aid of the bogotune program.

OPTIONS

HELP OPTIONS

The **-h** option prints the help message and exits.

The **-V** option prints the version number and exits.

The **-Q** (query) option prints bogofilter's configuration, i.e. registration parameters, parsing options, bogofilter directory, etc.

CLASSIFICATION OPTIONS

The **-p** (passthrough) option outputs the message with an X-Bogosity line at the end of the message header. This requires keeping the entire message in memory when it's read from stdin (or from a pipe or socket). If the message is read from a file that can be rewound, bogofilter will read it a second time.

The **-e** (embed) option tells bogofilter to exit with code 0 if the message can be classified, i.e. if there is not an error. Normally bogofilter uses different codes for spam, ham, and unsure classifications, but this simplifies using bogofilter with procmail or maildrop.

The **-t** (terse) option tells bogofilter to print an abbreviated spamicity message containing 1 letter and the score. Spam is indicated with "Y", ham by "N", and unsure by "U". Note: the formatting can be customized using the config file.

The **-T** provides an invariant terse mode for scripts to use. bogofilter will print an abbreviated spamicity message containing 1 letter and the score. Spam is indicated with "S", ham by "H", and unsure by "U".

The **-TT** provides an invariant terse mode for scripts to use. Bogofilter prints only the score and displays it to 16 significant digits.

The **-u** option tells bogofilter to register the message's text after classifying it as spam or non-spam. A spam message will be registered on the spamlist and a non-spam message on the goodlist. If the classification is "unsure", the message will not be registered. Effectively this option runs bogofilter with the **-s** or **-n** flag, as appropriate. Caution is urged in the use of this capability, as any classification errors bogofilter may make will be preserved and will accumulate until manually corrected with the **-Sn** and **-Ns** option combinations. Note this option causes the database to be opened for write access, which can entail massive slowdowns through lock contention and synchronous I/O operations.

The **-H** option tells bogofilter to not tag tokens from the header. This option is for testing, you should not use it in normal operation.

The **-M** option tells bogofilter to process its input as a mbox formatted file. If the **-v** or **-t** option is also given, a spamicity line will be printed for each message.

The **-b** (streaming bulk mode) option tells bogofilter to classify multiple objects whose names are read from stdin. If the **-v** or **-t** option is also given, bogofilter will print a line giving file name and classification information for each file. This is an alternative to **-B** which lists objects on the command line.

An object in this context shall be a maildir (autodetected), or if it's not a maildir, a single mail unless **-M** is given - in that case it's processed as mbox. (The Content-Length: header is not taken into account

currently.)

When reading mbox format, bogofilter relies on the empty line after a mail. If needed, **formail -es** will ensure this is the case.

The **-B** *object* ... (bulk mode) option tells bogofilter to classify multiple objects named on the command line. The objects may be filenames (for single messages), mailboxes (files with multiple messages), or directories (of maildir and MH format). If the **-v** or **-t** option is also given, bogofilter will print a line giving file name and classification information for each file. This is an alternative to **-b** which lists objects on stdin.

The **-R** option tells bogofilter to output an R data frame in text form on the standard output. See the section on integration with R, below, for further detail.

REGISTRATION OPTIONS

The **-s** option tells bogofilter to register the text presented as spam. The database is created if absent.

The **-n** option tells bogofilter to register the text presented as non-spam.

Bogofilter doesn't detect if a message registered twice. If you do this by accident, the token counts will off by 1 from what you really want and the corresponding spam scores will be slightly off. Given a large number of tokens and messages in the wordlist, this doesn't matter. The problem *can* be corrected by using the **-S** option or the **-N** option.

The **-S** option tells bogofilter to undo a prior registration of the same message as spam. If a message was incorrectly entered as spam by **-s** or **-u** and you want to remove it and enter it as non-spam, use **-Sn**. If **-S** is used for a message that wasn't registered as spam, the counts will still be decremented.

The **-N** option tells bogofilter to undo a prior registration of the same message as non-spam. If a message was incorrectly entered as non-spam by **-n** or **-u** and you want to remove it and enter it as spam, then use **-Ns**. If **-N** is used for a message that wasn't registered as non-spam, the counts will still be decremented.

GENERAL OPTIONS

The **-c** *filename* option tells bogofilter to read the config file named.

The **-C** option prevents bogofilter from reading configuration files.

The **-d** *dir* option allows you to set the directory for the database. See the ENVIRONMENT section for other directory setting options.

The **-k** *cachesize* option sets the cache size for the BerkeleyDB subsystem, in units of 1 MiB (1,048,576 bytes). Properly sizing the cache improves bogofilter's performance. The recommended size is one third of the size of the database file. You can run the bogotune script (in the tuning directory) to determine the recommended size.

The **-l** option writes an informational line to the system log each time bogofilter is run. The information logged depends on how bogofilter is run.

The **-L** *tag* option configures a tag which can be included in the information being logged by the **-l** option, but it requires a custom format that includes the %l string for now. This option implies **-l**.

The **-I** *filename* option tells bogofilter to read its input from the specified file, rather than from **stdin**.

The **-O** *filename* option tells bogofilter where to write its output in passthrough mode. Note that this only works when -p is explicitly given.

PARAMETER OPTIONS

The **-E** *value[,value]* option allows setting the sp-esf value and the ns-esf value. With two values, both sp-esf and ns-esf are set. If only one value is given, parameters are set as described in the note below.

The **-m** *value[,value][,value]* option allows setting the min-dev value and, optionally, the robs and robx values. With three values, min-dev, robs, and robx are all set. If fewer values are given, parameters are set as described in the note below.

The **-o** *value[,value]* option allows setting the spam-cutoff ham-cutoff values. With two values, both spam-cutoff and ham-cutoff are set. If only one value is given, parameters are set as described in the note below.

Note: All of these options allow fewer values to be provided. Values can be skipped by using just the comma delimiter, in which case the corresponding parameter(s) won't be changed. If only the first value is provided, then only the first parameter is set. Trailing values can be skipped, in which case the corresponding parameters won't be changed. Within the parameter list, spaces are not allowed after commas.

INFO OPTIONS

The **-v** option produces a report to standard output on bogofilter's analysis of the input. Each additional **v** will increase the verbosity of the output, up to a maximum of 4. With **-vv**, the report lists the tokens with highest deviation from a mean of 0.5 association with spam.

Option **-y date** can be used to override the current date when timestamping tokens. A value of zero (0) turns off timestamping.

The **-D** option redirects debug output to stdout.

The **-x** *flags* option allows setting of debug flags for printing debug information. See header file debug.h for the list of usable flags.

CONFIG FILE OPTIONS

Using GNU longopt **--** syntax, a config file's *name=value* statement becomes a command line's **--***option=value*. Use command **bogofilter --help** for a list of options and see bogofilter.cf.example for more info on them. For example to change the X-Bogosity header to "X-Spam-Header", use:

--*spam-header-name=X-Spam-Header*

ENVIRONMENT

Bogofilter uses a database directory, which can be set in the config file. If not set there, bogofilter will use the value of **BOGOFILTER_DIR**. Both can be overridden by the **-d** *dir* option. If none of that is available, bogofilter will use directory $HOME/.bogofilter.

CONFIGURATION

The bogofilter command line allows setting of many options that determine how bogofilter operates. File /usr/local/etc/bogofilter.cf can be used to set additional parameters that affect its operation. File /usr/local/etc/bogofilter.cf.example has samples of all of the parameters. Status and logging messages can be customized for each site.

RETURN VALUES

0 for spam; 1 for non-spam; 2 for unsure ; 3 for I/O or other errors.

If both **-p** and **-e** are used, the return values are: 0 for spam or non-spam; 3 for I/O or other errors.

Error 3 usually means that the wordlist file bogofilter wants to read at startup is missing or the hard disk has filled up in **-p** mode.

INTEGRATION WITH OTHER TOOLS

Use with procmail

The following recipe (a) spam-bins anything that bogofilter rates as spam, (b) registers the words in messages rated as spam as such, and (c) registers the words in messages rated as non-spam as such. With this in place, it will normally only be necessary for the user to intervene (with **-Ns** or **-Sn**) when bogofilter miscategorizes something.

```
# filter mail through bogofilter, tagging it as Ham, Spam, or Unsure,
# and updating the wordlist

:0fw
| bogofilter -u -e -p


# if bogofilter failed, return the mail to the queue;
# the MTA will retry to deliver it later
# 75 is the value for EX_TEMPFAIL in /usr/include/sysexits.h

:0e
{ EXITCODE=75 HOST }


# file the mail to spam-bogofilter if it's spam.

:0:
* ^X-Bogosity: Spam, tests=bogofilter
spam-bogofilter

# file the mail to unsure-bogofilter
# if it's neither ham nor spam.

:0:
* ^X-Bogosity: Unsure, tests=bogofilter
unsure-bogofilter

# With this recipe, you can train bogofilter starting with an empty
# wordlist.  Be sure to check your unsure-folder regularly, take the
# messages out of it, classify them as ham (or spam), and use them to
# train bogofilter.
```

The following procmail rule will take mail on stdin and save it to file spam if bogofilter thinks it's spam:

```
:0HB:
* ? bogofilter
spam
```

and this similar rule will also register the tokens in the mail according to the bogofilter classification:

```
:0HB:
* ? bogofilter -u
spam
```

If bogofilter fails (returning 3) the message will be treated as non-spam.

This one is for maildrop, it automatically defers the mail and retries later when the xfilter command fails, use this in your ~/.mailfilter:

```
xfilter "bogofilter -u -e -p"
if (/^X-Bogosity: Spam, tests=bogofilter/)
{
  to "spam-bogofilter"
}
```

The following .muttrc lines will create mutt macros for dispatching mail to bogofilter.

```
macro index d "<enter-command>unset wait_key\n\
<pipe-entry>bogofilter -n\n\
<enter-command>set wait_key\n\
<delete-message>" "delete message as non-spam"
macro index \ed "<enter-command>unset wait_key\n\
<pipe-entry>bogofilter -s\n\
<enter-command>set wait_key\n\
<delete-message>" "delete message as spam"
```

Integration with Mail Transport Agent (MTA)

1.

can also be integrated into an MTA to filter all incoming mail. While the specific implementation is MTA dependent, the general steps are as follows:

2.
bogofilter on the mail server

3.
the bogofilter databases with a spam and non-spam corpus. Since bogofilter will be serving a larger community, it is important to prime it with a representative set of messages.

4.
up the MTA to invoke bogofilter on each message. While this is an MTA specific step, you'll probably need to use the **-p**, **-u**, and **-e** options.

5.
up a mechanism for users to register spam/non-spam messages, as well as to correct mis-classifications. The most generic solution is to set up alias email addresses to which users bounce messages.

6.
the doc and contrib directories for more information.

Use of R to verify bogofilter's calculations

The -R option tells bogofilter to generate an R data frame. The data frame contains one row per token analyzed. Each such row contains the token, the sum of its database "good" and "spam" counts, the "good" count divided by the number of non-spam messages used to create the training database, the "spam" count divided by the spam message count, Robinson's f(w) for the token, the natural logs of (1 - f(w)) and f(w), and an indicator character (+ if the token's f(w) value exceeded the minimum deviation from 0.5, - if it didn't). There is one additional row at the end of the table that contains a label in the token field, followed by the number of words actually used (the ones with + indicators), Robinson's P, Q, S, s and x values and the minimum deviation.

The R data frame can be saved to a file and later read into an R session (see **the R project website**[5] for information about the mathematics package R). Provided with the bogofilter distribution is a simple R script (file bogo.R) that can be used to verify bogofilter's calculations. Instructions for its use are included in the script in the form of comments.

LOG MESSAGES

Bogofilter writes messages to the system log when the **-l** option is used. What is written depends on which other flags are used.

A classification run will generate (we are not showing the date and host part here):

bogofilter[1412]: X-Bogosity: Ham, spamicity=0.000227
bogofilter[1415]: X-Bogosity: Spam, spamicity=0.998918

Using **-u** to classify a message and update a wordlist will produce (one a single line):

bogofilter[1426]: X-Bogosity: Spam, spamicity=0.998918,
register -s, 329 words, 1 messages

Registering words (**-l** and **-s**, **-n**, **-S**, or **-N**) will produce:

bogofilter[1440]: register-n, 255 words, 1 messages

A registration run (using **-s**, **-n**, **-N**, or **-S**) will generate messages like:

bogofilter[17330]: register-n, 574 words, 3 messages
bogofilter[6244]: register-s, 1273 words, 4 messages

FILES

/usr/local/etc/bogofilter.cf
System configuration file.

~/.bogofilter.cf
User configuration file.

~/.bogofilter/wordlist.db
Combined list of good and spam tokens.

AUTHOR

Eric S. Raymond <esr@thyrsus.com>.
David Relson <relson@osagesoftware.com>.
Matthias Andree <matthias.andree@gmx.de>.
Greg Louis <glouis@dynamicro.on.ca>.

For updates, see the **bogofilter project page**[6].

SEE ALSO

bogolexer(1), bogotune(1), bogoupgrade(1), bogoutil(1)

NOTES

1. A Plan For Spam
   http://www.paulgraham.com/spam.html

2. Spam Detection
   http://radio-weblogs.com/0101454/stories/2002/09/16/spamDetection.html

3. A Statistical Approach to the Spam Problem
   http://www.linuxjournal.com/article/6467

4. Another improvement
   http://www.garyrobinson.net/2004/04/improved%5fchi.html

5. the R project website
   http://cran.r-project.org/

6. bogofilter project page
   http://bogofilter.sourceforge.net/