Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Bogofilter mata mejor (22694 lectures)

Per **Ricardo Galli Granada**, ***gallir*** (http://mnm.uib.es/gallir/)

Creado el 07/10/2002 18:59 modificado el 01/11/2002 12:38

Hace un tiempo escribí [1] que el spamassasin es muy bueno para detectar los spams, pero con tiempo de usarlo descubrí sus problemas. Entonces leí sobre métodos bayesianos de detección de spams. Y descubrí el bogofilter, que implementa este tipo de detección. En este artículo describo a grandes rasgos los fundamentos de funcionamiento y lo más importante, como configurarlo y "enseñarle" con un cuerpo (corpus) de spams que he ido coleccionado las últimas semanas.
Actualización: *Cambios en la versión 0.7.6. Explicado en la página 3 [2].*

Problemas del spamassasin

Actualización
Ya se encuentra en un estado bastante estable (ya no hay cambios constantes de formatos y opciones). Lo estoy usando hace menos de un mes, **hace más de una semana que no recibo ningún spam**. Y lo más importante, no ha dado ningún positivo en todo ese tiempo.
Actualizado el fichero de spams, hay más de mil. **Cuidado:** Si "entrenáis" al bogofilter con esos 1000 spams, tenéis que entrenarlo con el menos 3000 mensajes buenos (mantener la relación 1-3), sino dará muchos falsos positivos al principio.

El problema fundamental con el spamassasin es que está basado en el reconocimiento de patrones de texto usando reglas, lo que obliga a estar continuamente actualizando y agregando nuevas reglas para adaptarse a los cambios que introducen los *spammers* en sus mensajes. Además hay que elaborar reglas para cada lenguaje.

No sólo eso, para poder actualizar y mejorar la detección hay que introducir nuevas reglas, lo que hace que el proceso de análisis de cada mensaje sea aún más lento de lo que ya es actualmente, entre otras cosas porque está implementado en Perl y tiene que trabajar con muchas reglas.

Y no acaba aquí, en problema que creo que es el más grave, es que puede dar falsos positivos. Los falsos positivos son mensajes que no son spams, pero son considerados como tal por el filtro. Que de vez en cuando, como así sucede, llegue un spam que haya pasado el filtro (*falso negativo*), no pasa nada, pero los *falsos positivos* es lo peor que puede pasar, ya que puedes perder o dejar de leer mensajes importantes.

Me ha pasado bastante con amigos que me escriben en catalán, y con muchos mensajes de la lista de Bulma (las cabeceras “anti-abuse” que pone el mailman son consideradas como cabeceras de spammers por el spamassassin). Así que me decidí a probar otras cosas, el spamassasin aunqueme convenció en un principio, al poco tiempo mostró sus problemas.

Un plan para el spam

Durante un tiempo estuve investigando que métodos había para detección de spams, pensaba que la lógica difusa podía ser una solución, hasta que encontré el ensayo de Paul Graham, "A Plan for Spam[3]". En dicho ensayo Paul Graham opina que los métodos basados en reglas no sirven para detectar spams, no sólo por los problemas mencionados anteriormente, sino porque sería muy fácil para los *spammers* encontrar los trucos para saltarse las reglas de filtrados, ya que son de dominio público y común para todas las instalaciones.

La solución, según él (y yo también :-) pasa por usar métodos que "aprendan" de los mensajes que recibe cada usuario y generar su propia base de datos de palabras. Esta base de datos sirve para calcular las probabilidades combinadas de que un mensaje sea o no spam.

El método "bayesiano" que propone es muy interesante, aunque ha recibido críticas y sugerencias,[4] ya que en éste caso el filtro se adaptaría automáticamente al idioma y tipos de mensajes que recibe cada uno.

El problema fundamental de ésta aproximación es que hay que *entrenar* inicialmente al programa con un conjunto relativamente de mensajes spams y otros válidos para que arme su base de datos inicial. A partir de allí el programa puede aplicar los métodos bayesianos y usar esos mismos mensajes, ya clasificados como spam o no, para realimentar la base de datos.

Fue así que encontré una implementación, levemente mejorada, de éste método, el **bogofilter**.

Bogofilter

El bogofilter[5] es una implementación, muy eficiente, en C y con db2 para almacenar las palabras y sus apariciones en mensajes programado por Eric Raymond. Al día que estoy escribiendo ésto, el bogofilter está en su versión 0.7.4cvs1004 y a pesar que sufrió bastantes cambios "molestos" en los formatos utilizados (que obligaban a regenerar la base de datos por cada actualización), desde hace un par de semanas que no veo cambios de éste tipo, por lo que se puede estar casi tranquilo que no habrá trabajos "complejos" de migración hasta su versión estable.

Funcionamiento

Bogofilter mantiene un par de bases de datos en el directorio *$HOME/.bogofilter/*:

- goodlist.db
- spamlist.db

Cada una de ellas mantiene una lista de "tokens" (palabras) junto con la cantidad de veces que esa palabra ha aparecido en mensajes válidos o spams. Esos números son usados para calcular la probabilidad de que el mensaje sea un spam.

Una vez que se han calculado las probabilidades, se usan aquellas más alejadas de la media para combinarlas usando el Teorema de Bayes de probabilidades combinadas. Si la probabilidad combinada es mayor que 0.9, bogofilter retorna 1, caso contrario retorna 0.

La fiabilidad del bogofilter depende exclusivamente de la cantidad de palabras que tenga en su base de datos, mientras más tenga y mayor se la cantidad de aparciciones de cada palabra en un mensaje válido o spam, mejores serán sus resultados.

Si sólo le enseñanmos al principio cuales son mensajes válidos, no podrá detectar los spams. Al contrario, si sólo le "enseñamos" spams, considerará a muchos mensajes válidos como spams (falsos positivos...). O sea, el aprendizaje inicial es importantísimo, y nos ahorrará mucho trabajo de mantenimiento de la base de datos.

En resumen, lo que hay que hacer, además de instalar el programa (*apt-get install bogofilter* :-), son los siguientes pasos:

1. Entrenarlo con un conjunto relativamente grande de mensajes válidos que tengamos almacenados.
2. Entrenarlo con un conjunto grande de spams (no os preocupéis, os doy una lista de más de 700 spams).
3. Configurar el .procmailrc.

4. Seguimiento y mantenimiento de los primeros días.

Entrenarlo con mensajes válidos

Este paso es muy importante, caso contrario puede generar falsos positivos. Lo mejor en estos casos es entrenarlo con los mensajes que tengamos almacenados en nuestro cliente de correo. Yo aconsejaría que por lo menos lo hagáis con unos 1000 mensajes, y cuanto más, mejor.

Si tenemos los mensajes almacenados en formato mbox, basta con hacer lo siguiente:

```
bogofilter -n < fichero_mbox
```

Si por el contrario lo tenemos en formato maildir, y debido a que el estándar de éste formato quita las líneas de "From" de inicio de mensaje, el bogofilter no es capaz de separar y contar los mensajes, por lo que hay que llamarlo por cada mensaje almacenado. Es muy fácil hacerlo con un pequeño script:

```
for f in directorio_maildir/*
  do
  bogofilter -n < $f
  done
```

Fijaros que luego de este paso, debería haber creado el fichero *$HOME/.bogofilter/goodlist.db*.

Entrenarlo con spams

El siguiente paso será entrenarlo con spams. En bulma.net/~gallir/BULMA/spams.txt.gz[6] tenéis una lista de más de 700 mensajes de spams (que iré actualizando periódicamente). Cuando hayáis bajado esta lista, la usaréis para entrenar al bogofilter de la siguiente manera:

```
zcat spams.txt.gz | bogofilter -S
```

Luego de realizado este paso, debería que haber creado el fichero *$HOME/.bogofilter/spamlist.db*.

Ahora ya lo tenemos casi todo listo, sólo falta activarlo en el procmail.

Configuración del .procmailrc

A continuación hay que poner las llamadas en el procmail para hacer que se llame el bogofilter por cada mensaje que nos llega. Una vez que clasificado como spam, mover el mensaje a otro fichero o carpeta (en el ejemplo lo muevo a *$HOME/mail/spams*).

Además de ello, lo que haremos es realimentar la base de datos con cada mensaje que llega, por lo que el sistema irá aprendiendo con el tiempo, y veréis que a las pocas semanas ya no necesita casi mantenimiento, si es que lo necesita...:

```
# fichero $HOME/.procmailrc
:0HB
* ? bogofilter
{
   # Es un spam
        # Realimentamos la base de datos
        :0HBc
        | bogofilter -s
```

```
        # lo movemos al fichero $HOME/mail/spams
        :0
        mail/spams
}

:0EHBc
# es un mensaje válido, realimentamos la base de datos
| bogofilter -n
```

NOTA: Ver la página 3, con las últimas versiones se simplifican bastante las reglas del .procmailrc.

Seguimiento y mantenimiento

Aunque según mi experiencia, con el entrenamiento inicial usando un buen conjunto de mensajes, es suficiente para que el bogofilter casi no falle, hay que ir con cuidado al menos los primeros días para verificar que no haya dado positivos y también para marcar como spams aquellos que los haya considerado válidos.

Como hemos puesto en el .procmailrc que la base de datos sea actualizada con cada mensaje, ahora no basta con marcar como spam un mensaje que haya considerado válido (y viceversa), sino que además hay que "volver atrás" los incrementos que se hicieron para las palabras del mensaje. Para eso existen las versiones **-N** y **-S**.

En ambos casos lo que tenemos que hacer es grabar el mensaje en un fichero de texto y ejecutar con la opción que corresponda, -S si es un spam y -N si es un falso positivo.

Marcar como spam un mensaje

Primero grabamos el spam que no ha sido detectado, por ejemplo en el fichero s.txt y luego hacemos:

```
bogofilter -S < s.txt
```

Marcar como válido un falso positivo

Grabamos ese mensaje (que debería estar en el fichero mbox *$HOME/mail/spams* si habéis seguido el ejemplo) como n.txt y hacemos:

```
bogofilter -N < n.txt
```

En la siguiente página, **los cambios en la última versión disponible: 7.6**

Cambios en la versión 0.7.6

Si disponéis de la versión 0.7.6 o superior, se han introducido un par de cambios interesantes:

1. Actualización automática de la base de datos.
2. Algoritmo de Robinson alternativo.

Actualización automática de la base de datos

Han agregado la opción **-u** que indica al bogofilter que actualice directamente la base de datos de acuerdo a la clasificación que se haga del mensaje. La ventaja principal es que simplifica enormemente el .procmailrc, ya que no hay que hacer varias llamadas.

Ahora es suficiente con:

```
# fichero $HOME/.procmailrc
:0HB
```

```
* ? bogofilter -u
mail/spams
```

Algoritmo de Robinson

Gary Robinson criticó y propuso un algoritmo alternativo[4]. Los desarrolladores de bogofilter no se han quedado atrás y también han implementado dicho algoritmo. Para usarlo hay que especificar la **opción -r**.

Yo llevo casi una semana probando los dos, con los mismos mensajes pero en ordenadores distintos. Se han comportado igual, salvo en dos ocasiones, donde el algoritmo de Robinson sí los clasificó correctamente:

1. Mensaje de las Debian Weekly News: el algoritmo tradicional lo clasificó como spam (falso positivo). El de Robinson lo clasificó correctamente como no-spam.
2. Un mensaje de spam en castellano: El tradicional lo dejó pasar (falso negativo). El de Robinson lo clasificó correctamente como spam.

Se puede decir que ambos funcionan muy bien, pero el algoritmo de Robinson se comporta *un poco mejor* que el tradicional. Aunque la diferencia puede haber sido accidental.

Conclusiones

Yo estoy muy contento en como funciona, aunque al principio fue bastante molesto porque Eric Raymond cambió varias veces el formato de la base de datos y los nombres de ficheros usados[7], ahora parece muy estable y no he tenido problemas prácticamente con falsos positivos. Al principio también tenía problemas en separar las palabras con ASCII mayores a 127 (parece un defecto de los programadores norte-americanos, sólo piensan en inglés), ahora ya está solucionado (por el mantenedor del paquete en Debian, Clint Adams).

Además me estoy haciendo mi propia base de datos de spams (el spams.txt.gz que mencioné anteriormente), más los mensajes válidos guardados, aunque cambien el formato en unos pocos segundos podré volver a mi estado actual (y actualizaré este artículo, no sufráis :-).

Nada más que decir, sólo que os lo recomiendo. Además en el futuro próximo vendrá con muchas mejoras, sobre todo para servidores de muchas cuentas de correo...

NOTA: En las últimas versiones, las opciones para clasificar son -s (-S) y -n (-N) para el spam y válidos respectivamente. En versiones anteriores es -s (-S) y -h (-H). Verificad vuestra versión. La "h" es de *ham*: jamón, para diferenciarlo del "spam", que era una marca de carne enlatada. Cosas del ESR...

Lista de enlaces de este artículo:

1. http://bulma.net/./body.phtml?nIdNoticia=1389
2. http://bulma.net/body.phtml?nIdNoticia=1537&nIdPage=3
3. http://www.paulgraham.com/spam.html
4. http://radio.weblogs.com/0101454/stories/2002/09/16/spamDetection.html
5. http://bogofilter.sourceforge.net/
6. http://bulma.net/~gallir/BULMA/spams.txt.gz
7. http://lists.debian.org/debian-devel/2002/debian-devel-200209/msg02148.html

---

E-mail del autor: gallir _ARROBA_ uib.es
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1537