

Jacek Śliwerski
Instytut Informatyki, Uniwersytet Wrocławski

Statystyczny filtr antyspamowy

3 czerwca 2003 roku

Streszczenie

Program pocztowy Elmo wyposażony jest w statystyczny filtr antyspamowy, chroniący użytkownika przed niechcianą pocztą. Dokument ten opisuje działanie filtra, możliwości jego konfiguracji oraz omawia dokładnie zasadę jego działania.

Tekst zawiera również krótkie omówienie innych metod filtrowania spamu, lecz moje subiektywne podejście może wypaczać ocenę możliwości innych algorytmów.

1. Tło

Niezamówione, nieoczekiwane i niechciane przesyłki dostarczane są codziennie milionom ludzi na całym świecie za pomocą poczty zarówno tradycyjnej, jak i elektronicznej. Problem stał się na tyle poważny, że w wielu krajach proceder taki sankcjonowany jest prawem, a złamanie zakazu traktowane jest jak wykroczenie.

„A tu pospolitość skrzeczy...” powiedziała by poeta obserwując rodzime rozwiązania prawne tej kwestii (choć chyba niewiele się różnią od zachodnich). Udowodnienie komuś, że złamał przepisy dotyczące wysyłania ofert handlowych pocztą elektroniczną może być niewykonalne. Użytkownicy darmowych kont pocztowych są zasypywani spamem zgodnie z obowiązującym prawem. Najlepsze więc wydaje się postępowanie zgodnie z zasadą: *„Umiesz liczyć, licz na siebie”*.

2. Metody alternatywne

Metod pozbywania się spamu jest bardzo wiele. Trzeba sobie jednak zdawać sprawę z tego, że jedynym w 100% skutecznym algorytmem jest ręczne usuwanie spamu.

2.1. Białe, czarne listy

Jaki jest pierwszy sposób, który przychodzi do głowy? Każdy list ma nadawcę. Można więc stworzyć „czarną listę” osób, które wysyłają spam i automatycznie odrzucać przesyłki przychodzące z takich adresów. W Internecie znajdują się bazy danych zawierające zbiory adresów, z których wysyłano spam. Przykładem może być serwis PolSpam, który zbiera adresy osób wysyłających spam.

Choć na pierwszy rzut oka rozwiązanie to może wydawać się kuszące, nie spełnia ono jednak swojej roli. Należy zwrócić uwagę, że w bazie danych może się pojawić każdy adres email – również

mój. Jeśli dostatecznie dużo osób doniesie do PolSpamu, że jestem spamerem, mój adres znajdzie się w bazie danych i moje listy mogą nie dojść nawet do moich znajomych! Dużo zależy również od oprogramowania wykorzystującego bazę danych PolSpamu. Jeśli filtr sprawdza tylko zawartość pola **From** z nagłówka wiadomości, to może to nie odnieść żadnego skutku, ponieważ w polu tym może się znajdować dowolny tekst, również adres odbiorcy. Może się to wydawać nieprawdopodobne, ale spamer może wysłać do mnie wiadomość, która (z punktu widzenia takiego filtra) została wysłana przeze mnie.

Częściowym rozwiązaniem problemu jest stworzenie białej listy, na której znajdują się adresy, z których na pewno nie przychodzi spam. Tylko pozostałe wiadomości przechodzą wówczas proces weryfikacji poprawności adresu nadawcy. Nie zmienia to jednak faktu, że brak jest logicznego uzasadnienia poprawności działania takiego filtra.

2.2. Indywidualne cechy spamu

Metodę czarnych list można by jednak uogólnić. Dlaczego mielibyśmy sprawdzać tylko adres, z którego przychodzi wiadomość, a nie pozostałe pola nagłówka, albo nawet treść listu. Spamer powinien w jakiś sposób starać się przyciągnąć uwagę odbiorcy do treści, które ma mu do zakomunikowania. Dobrym sposobem wydaje się być znajdowanie pewnych wzorców występujących w tekście. Kandydaci na słowa-klucze, to: sex, oferta, darmo. Również duża liczba wykrzykników może sugerować, że list jest spamem.

Skuteczność tego rozwiązania jest z pewnością znacznie większa. Spamer może podmienić adres w polu **From**, ale nie może zmienić tytułu, ani treści listu – to nadal musi być oferta, która przyciągnie potencjalnego klienta.

Niestety każdy człowiek odróżnia spam na podstawie innych cech. Czy seksuolog może sobie pozwolić na odfiltrowywanie listów na podstawie zawierania słowa sex? Czy handlowiec może dyskryminować listy, które w tytule mają słowo oferta? Czy bankowiec zdecyduje się usunąć list, który zawiera 12 znaków \$? Wniosek płynie stąd taki, że każdy użytkownik powinien sam ustalić kryteria eliminujące nieporządane przesyłki, albo zgodzić się na zastosowanie pewnych ogólnych wskaźników, które być może okażą się nieadekwatne. Drugą zasadniczą wadą tego rozwiązania jest fakt, że spam ewoluuje. Jeśli zdecydujemy się odfiltrowywać listy zawierające słowo „friko”, to spamerzy zaczną wysyłać listy, w których „friko” zastąpione zostanie ciągiem „f.r.i.k.o”. Zmusza to użytkownika do ciągłej aktualizacji reguł filtra.

3. Filtr statystyczny

Zanim przejdę do teoretycznych i technicznych szczegółów algorytmu chciałbym po krótko nakreślić jakie idee przyświecały jego twórcy oraz krótko uzasadnić jego poprawność.

Algorytm zaimplementowany w Elmo opisany został przez Paula Grahama w jego artykule p.t. „A Plan for Spam” [1] oraz w jego kontynuacji zatytułowanej „Better Bayesian Filtering” [2].

3.1. Idea

Załóżmy, że użytkownik posiada swoją pocztę skatalogowaną w dwóch korpusach. Pierwszy zawiera poprawną korespondencję, a drugi zawiera spam. Pomysł polega na przeprowadzeniu analizy

statystycznej wystąpień słów w obydwu korpusach. Chodzi o to, aby móc stwierdzić, które słowa są „dowodami” bycia spamem, a które słowa są „dowodami” bycia poprawną wiadomością. Każdy przychodzący list będzie potem analizowany pod kątem zawierania odpowiednich „dowodów”. Algorytm pozwoli nam obliczyć prawdopodobieństwo, z jakim filtr może stwierdzić, że dana wiadomość jest spamem. Ustalimy następnie kryterium dyskryminacji, które odrzuci listy o dużym prawdopodobieństwie bycia spamem.

Taki filtr automatycznie dostosuje się do indywidualnych predyspozycji użytkownika. Wiadomości od moich znajomych ocechowane są ich adresami email, imionami i nazwiskami. W listach tych jest poruszana tematyka związana z moimi zainteresowaniami. Wszystkie takie słowa zostaną uznane za „dowody” bycia poprawną wiadomością. Jako użytkownik filtru oszczędzam sobie również analizy cech spamu. Filtr sam zdecyduje jakie cechy wiadomości są dobrymi „dowodami” na bycie spamem.

3.2. Implementacje

Istnieje bardzo wiele różnych implementacji algorytmu Paula Grahama. Najpopularniejszą jest chyba program bogofilter[3]. Większość tego typu oprogramowania to osobne, uniwersalne programy, które należy zintegrować z własnym programem pocztowym. Istnieje również garstka programów pocztowych zawierających „wbudowaną” obsługę filtru statystycznego. Przykładem może być Mozilla Mail&News, albo Elmo. Przy wyborze odpowiedniego programu należy zwrócić uwagę na jego wymagania. Radzę unikać implementacji wymagających posiadania relacyjnej bazy danych (SQL). Algorytm wymaga przechowywania dużej ilości danych na temat wiadomości, ale zaprzęganie do tego zadania tak potężnego silnika jak SQL wydaje się być nieuzasadnione.

3.3. Wyniki

Opis algorytmu może być zachęcający, ale dużo ciekawsze są wyniki, jakie otrzymał Paul podczas pracy nad swoim algorytmem. Od 10 grudnia 2002 roku do 10 stycznia 2003 roku otrzymał on 1750 listów kwalifikujących się jako spam. Filtr nie odrzucił 4 z nich. Oznacza to skuteczność na poziomie 99.75%. Dwa z nich zawierały słowa, które często występują w poprawnej poczcie Paula. Trzeci trafił do niego przez skrypt cgi, który umożliwiał wysłanie listu do dowolnego odbiorcy. Czwarty zawierał tylko informację, że ktoś właśnie skończył pracę nad swoją stroną i zaprasza do odwiedzenia. Oczywiście podany odnośnik prowadził do strony pornograficznej.

W ciągu tego samego miesiąca algorytm trzykrotnie zakwalifikował poprawny list jako spam. Trzeba jednak przybliżyć, jak wyglądały listy, które zostały odfiltrowane. Dwa z nich zawierały oferty firm, z których usług Paul skorzystał wcześniej. Właściwie możnaby je zakwalifikować jako spam, ponieważ nie życzył sobie ich otrzymać, ale ponieważ nigdy nie kasował takich wiadomości, to uznał, że w tych przypadkach algorytm się pomylił. Trzeci list wysłany został z Egiptu i był w całości napisany kapitalikami. Paul opisuje również późniejsze przypadki błędnych odpowiedzi, których ogółem było 5. Przy ogólnej liczbie 7740 wiadomości daje to błąd rzędu 0.06%.

Trzeba tu jednak nadmienić, że porównywanie wydajności (i odpowiedniości) różnych filtrów jest bardzo trudne. Implementacje mogą różnić się pewnymi szczegółami, które mogą być krytyczne dla ogólnego wyniku działania filtru. Duża skuteczność filtru wynikała również z bardzo dużych rozmiarów obydwu korpusów (ok. 4 tys. wiadomości w każdym).

4. Z punktu widzenia użytkownika

Pierwsza rzecz, jaka przyszła mi na myśl jeszcze przed rozpoczęciem implementacji, to fakt, że filtr powinien być przezroczysty dla osoby posługującej się nim. Paul sugerował w swoim artykule, aby program pocztowy miał dwa przyciski do usuwania listów. Jeden z nich powinien być oznaczony „usuń jako spam”. Tak też jest w Elmo.

Aby uaktywnić filtr antyspamowy należy w definicji skrzynki (zmiennej `mailbox`) ustawić polu¹ `protect` wartość `yes`. Aby spam nie był automatycznie usuwany należy jeszcze do pola `spam` przypisać nazwę skrzynki, do której spam będzie przesuwany. Przykładowa definicja skrzynki:

```
set mailbox {
  name:      'moja poczta'
  root:      ~/mail
  trash:     trash
  spam:      spam
  protect:   yes
}
```

Powyższa definicja sprawi, że Elmo będzie automatycznie oceniało listy. Wszystkie funkcje modułu `bayes` są jednak nieaktywne w skrzynkach `trash`, `spam`, `sent` oraz `drafts`. Ocena listu wyświetlana jest zgodnie z definicją zmiennej `line_format`. W miejsce „%#” podstawiana jest jedna z trzech wartości:

- - list jest poprawny,
- ? - list nie został oceniony,
- # - list jest spamem.

Wywołanie funkcji `folder_flush` (domyślnie związanej z klawiszem `$`) spowoduje przeniesienie spamu do wskazanej skrzynki (lub usunięcie) oraz aktualizację tablic hashujących przechowujących liczbę wystąpień słów w korpusach z poprawną pocztą i spamem. Aktualizacja następuje również przy opuszczaniu skrzynki oraz przed zakończeniem działania programu. Listy ocenione jako poprawne otrzymują wówczas flagę „`Legitimate`” co zapobiega ich ponownej ocenie².

Jeśli filtr zakwalifikuje list będący spamem jako poprawną wiadomość, należy ją usunąć przy pomocy funkcji `folder_spam_delete`. Jeśli zaś list został niesłusznie uznany za spam należy na nim uruchomić funkcję `folder_spam_is_not`.

5. Teoria

Algorytm Paula Grahama nie jest poparty żadnym aparatem matematycznym. Postaram się jednak odtworzyć rozumowanie, jakim posłużył się autor algorytmu podczas jego tworzenia.

¹Więcej informacji na temat ustawiania zmiennych znajduje się w dokumentacji dołączanej do dystrybucji programu oraz w przykładowych plikach konfiguracyjnych.

²Jest to rozszerzenie formatu `maildir`, w którym Elmo standardowo trzyma pocztę. Pozostałe programy pocztowe powinny ignorować tę flagę.

Paul zaproponował, aby dla każdego słowa występującego w jednym z korpusów obliczyć pewną wartość, która pozwala zdecydować, jak dalece słowo jest „dowodem” bycia spamem, albo „dowodem” bycia wiadomością poprawną.

Wprowadźmy następujące oznaczenia:

- w – słowo,
- $S(w)$ – liczba wystąpień w w korpusie ze spamem,
- $L(w)$ – liczba wystąpień w w korpusie z pocztą poprawną,
- m – liczba wiadomości w korpusie ze spamem,
- n – liczba wiadomości w korpusie z pocztą poprawną,
- X – zdarzenie: wiadomość M jest spamem,
- Y – zdarzenie: $w \in M$.

Gdyby prawdziwe były następujące równości kombinatoryczne:

$$\mathcal{P}(X \cap Y) = \frac{S(w)}{m} \quad (1)$$

$$\mathcal{P}(Y) = \frac{S(w)}{m} + \frac{L(w)}{n} \quad (2)$$

to korzystając ze wzoru na prawdopodobieństwo warunkowe:

$$\mathcal{P}(X|Y) = \frac{\mathcal{P}(X \cap Y)}{\mathcal{P}(Y)} \quad (3)$$

otrzymalibyśmy pierwszą część wzoru

$$p(w) = \begin{cases} \frac{\frac{S(w)}{m}}{\frac{L(w)}{n} + \frac{S(w)}{m}} & , \quad S(w) + L(w) > 5 \\ 0.4 & , \quad S(w) + L(w) \leq 5 \end{cases} \quad (4)$$

Niestety zarówno wzór (1) jak i wzór (2) nie określają prawdopodobieństwa, ponieważ obydwie wartości mogą być większe od 1. Dodatkowo wzór (2) mógłby być prawdziwy jedynie w sytuacji, gdyby korpusy były rozłączne. Algorytm wymusza (przy pomocy funkcji minimum i maksimum), aby obliczona wartość $p(w)$ znalazła się w przedziale $[0.01, 0.99]$.

Z tego powodu nie będą wartości $p(w)$ nazywał prawdopodobieństwem. Jest to jednak dość dobry współczynnik stanowiący o tym, czy słowo w jest „dowodem” na bycie spamem (wartości zbliżone do 1), albo na bycie wiadomością poprawną (wartości bliskie 0). Zwróćmy uwagę na to, że duże wartości wyrażenia $\frac{S(w)}{m}$ oznaczają częste występowanie słowa w w listach ze spamem, a duże wartości wyrażenia $\frac{L(w)}{n}$ oznacza częste występowanie słowa w w poprawnych wiadomościach. Wzory (1) i (2) można by poprawić poprzez wstawienie do mianownika ilości listów z danego korpusu, w którym dane słowo rzeczywiście wystąpiło, ale osłabiłoby to znacznie siłę dyskryminacji $p(w)$, ponieważ nie uwzględniałoby wielokrotnych wystąpień słowa w jednym liście.

Algorytm wybiera z wiadomości M 15 słów najbardziej odległych od arbitralnie przyjętej wartości 0.5, a następnie oblicza dla nich wartość funkcji $f(M)$ w następujący sposób:

$$f(M) = \frac{\prod_{i=1}^{15} p(w_i)}{\prod_{i=1}^{15} p(w_i) + \prod_{i=1}^{15} (1 - p(w_i))} \quad (5)$$

Kryterium dyskryminacji to $f(M) > 0.9$.

Wzór (5) jest prawdziwy przy założeniu, że prawdopodobieństwa $p(w_i)$ są parami niezależne, a $\mathcal{P}(X) = 0.5$. Oczywiście nie jest to prawda w przypadku słów znajdujących się w korespondencji choćby ze względu na kontekst.

6. Implementacja filtru w Elmo

Implementacja algorytmu znajduje się w 4 plikach źródłowych. Pliki `token.c` i `token.h` definiują funkcje operujące na tokenach, to znaczy rozbijają ciąg znaków na słowa zdefiniowane odpowiednim wyrażeniem regularnym. Pliki `bayes.c` oraz `bayes.h` zawierają implementację samego filtru.

Moduł `bayes` jest mieszaną implementacją algorytmu prostego (zaproponowanego w pierwszej pracy Paula) i usprawnionego z pewnymi poprawkami, które sam wprowadziłem. Moduł ma bardzo wiele współczynników, które można modyfikować. Wszystkie dane, które będą poniżej wymieniał są makrodefinicjami w pliku `bayes.c` i umożliwiają dostrojenie algorytmu do swoich potrzeb.

Usprawnienia uwzględnione w Elmo:

- tokeny są case-sensitive: "Free" i "free" to nie są te same tokeny;
- adresy nie są rozdzielane: "ala@kot.org" jest pojedynczym tokenem, kropki w środku słowa są jego częścią;
- tokeny często występujące wyłącznie w korpusie ze spamem są bardziej interesujące w sensie algorytmu niż te występujące mniej niż 10 razy (ale również wyłącznie w korpusie ze spamem).

Usprawnienia jeszcze nie uwzględnione:

- brak strukturalizacji i degeneracji tokenów.

Moje własne zmiany w algorytmie:

- z nagłówka skanowane są tylko wybrane części i to dopiero po rozkodowaniu – Received, Sender, From, To, Subject;
- ograniczenie maksymalnej długości słowa do 60 znaków.

Jeśli chodzi o skanowanie nagłówków, to pozwoliłem sobie na zignorowaniu pozostałych nagłówków, gdyż zawierają losowe dane, które tylko zaśmiecają tablice hashujące, a niczego nie wnoszą. Aby moduł działał szybciej, prawie wszystkie operacje są wykonywane na liczbach całkowitych. A to pełna lista definicji sterujących zachowaniem modułu:

PROB_CERTAIN służy do sterowania dokładnością prawdopodobieństwa. Domyślna wartość 10000 oznacza, że pod uwagę będą brane cztery cyfry po przecinku.

PROB_NEUTRAL określa prawdopodobieństwo dla słowa, które pojawiło się w obydwu korpusach mniej niż **PROB_MIN_OCCUR** razy.

PROB_LIMIT to prawdopodobieństwo powyżej którego listy będą kwalifikowane jako spam.

PROB_HALF określa które słowa będą bardziej interesujące z punktu widzenia algorytmu. Wartości bliższe 0 będą preferowały słowa wskazujące na poprawność listu, a bliższe 1 będą preferowały słowa wskazujące na spam. Ta liczba powinna być z zakresu 0.49 – 0.51.

PROB_MIN określa minimalną wartość prawdopodobieństwa oraz prawdopodobieństwo słowa, które występuje wyłącznie w korpusie z poprawną pocztą.

PROB_MAX określa prawdopodobieństwo słowa, które występuje wyłącznie w korpusie ze spamem, ale mniej niż 10 razy.

PROB_MAX_MAX określa maksymalne prawdopodobieństwo słowa oraz wartość prawdopodobieństwa dla słowa, które występuje wyłącznie w korpusie ze spamem, ale więcej niż 10 razy.

WORD_MAX_LEN to maksymalna długość tokenu.

HEAP_SIZE oznacza ilość słów branych pod uwagę przy obliczaniu prawdopodobieństwa. Większa ilość słów będzie wpływać niekorzystnie na rozpoznawanie długich listów, ale może zmniejszyć prawdopodobieństwo pomyłki na niekorzyść poprawnego listu. Ta liczba musi być potęgą dwójki.

Literatura

- [1] <http://www.paulgraham.com/spam.html>
- [2] <http://www.paulgraham.com/better.html>
- [3] <http://bogofilter.sourceforge.net>
- [4] <http://www.mathpages.com/home/kmath267.htm>
- [5] <http://radio.weblogs.com/0101454/stories/2002/09./16/spamDetection.html>