Filtrage de SPAM par méthodes probabilistes

Les courriers électroniques publicitaires polluent les boîtes des internautes depuis de nombreuses années et le problème semble empirer de jour en jour. De nombreuses méthodes ont été développées pour lutter contre le SPAM, avec assez peu de succès.

La situation évolue pourtant de façon positive avec l'apparition d'une nouvelle catégorie de méthodes, le filtrage par approche probabiliste. L'objectif de cet article est de présenter les arguments mathématiques qui justifient la reconnaissance de SPAM par méthodes probabilistes, en détaillant les algorithmes utilisés.

Introduction

Un SPAM est un courrier électronique non sollicité envoyé de façon automatisée. Je crois pouvoir affirmer au nom de millions d'internautes que c'est un véritable fléau. Les Québécois résumant l'avis général en désignant le SPAM par un joli nom, le **pourriel**, contraction de courriel pourri (ou de courriel poubelle). Quand on possède une adresse électronique relativement vieille (quelques années), on peut facilement en recevoir une centaine par jour (certaines adresses connues reçoivent plusieurs centaines de SPAM par jour), le courrier intéressant étant noyé dans cette masse. Les non anglophones ont pour l'instant un peu de chance car la majorité des SPAM est rédigée en anglais : si on reçoit essentiellement du courrier en français (par exemple), il est en effet relativement facile de distinguer rapidement les *emails* potentiellement intéressants au milieu des ordures. Pour les anglophones, la gêne est largement supérieure. Bien entendu, ceci n'est que temporaire, les SPAM en français étant de plus en plus nombreux.

De plus, le SPAM coûte cher (aux destinataires, bien sûr), essentiellement en temps perdu à effacer les pourriels. Diverses évaluations de ce coût ont été réalisées. Une étude de *Ferris Research* [1], effectuée en décembre 2002, proposait par exemple une estimation de 8.9 milliards de dollars de perte par an pour les entreprises américaines (2.5 milliards de dollars de perte pour les entreprises européennes), le coût pour les ISP US et européens était estimé à 500 millions de dollars par an (en grande partie à cause du gaspillage de bande passante).

En revanche, le coût de l'envoi est tellement faible (une étude du *ePrivacy Group* datant de début 2003 [2] propose une estimation de 100 dollars par million de messages) qu'un taux de retour minuscule suffit à assurer des bénéfices aux *spammers* (les expéditeurs de SPAM). Le taux de 0.001 % est régulièrement avancé. Il suffit donc qu'un naïf (pour ne pas employer de mots plus forts) achète une boîte de pilules sensées lui donner un pénis gigantesque pour que l'envoi de 100 000 *emails* vantant les mérites des fameuses pilules soit rentable ! Par comparaison, l'étude du *ePrivacy Group* indique qu'il faut un taux d'au moins 5 % de réponses positives pour rentabiliser un envoi massif traditionnel. Voilà pourquoi nos boîtes électroniques sont souvent beaucoup plus encombrées que nos boîtes aux lettres classiques.

Le problème du SPAM est tel que de nombreuses solutions techniques ont été et sont encore développées aujourd'hui pour le contrer, c'est-à-dire pour filtrer les *emails* indésirables. Je vais présenter dans cet article une des méthodes les plus récentes et les plus efficaces, le filtrage probabiliste, aussi appelé filtrage Bayésien, très à la mode depuis le succès de l'article de Paul Graham, *A Plan for Spam* [3].

Le filtrage de SPAM

Présentation du problème

Le filtrage de SPAM est un cas particulier du problème général de la reconnaissance de forme, au sens large du terme. On regroupe en effet sous une même dénomination l'ensemble des problèmes dans lesquels on cherche à produire un dispositif automatique (un programme, en général) capable de ranger des observations dans plusieurs classes définies au préalable. Par classe j'entends un groupe d'objets homogènes relativement à une propriété qui m'intéresse. Pour fixer les idées, voici quelques exemples de problèmes de reconnaissance de forme :

- le système Graffiti (r) des Palms (tm) [4] : il s'agit d'un alphabet simplifié qu'on utilise pour écrire sur un assistant personnel. Le système d'exploitation des Palms contient un programme qui associe, à un tracé produit par l'utilisateur, une lettre (ou une commande). Les classes étudiées sont celles correspondant aux commandes que l'assistant doit être capable de reconnaître (les lettres de l'alphabet, les chiffres et des commandes comme l'effacement du dernier caractère saisi) ;
- certains téléphones portables proposent une commande vocale : le téléphone reconnaît la voix de son maître ! Les classes sont en général les numéros de téléphone. On associe un mot à un numéro dans le répertoire et le téléphone compose automatiquement le numéro quand on répète le mot ;
- les logiciels d'OCR traduisent un document scanné en un texte, ils reconnaissent donc les lettres imprimées (qui sont les classes du problème).

On peut être surpris de l'utilisation du terme "forme" pour décrire un SPAM. En fait, les éléments d'un *email* sont tout aussi organisés que les niveaux de gris d'un document scanné ou que les fréquences d'un enregistrement audio. Le principe de la reconnaissance de forme est simplement d'observer les caractéristiques d'un objet (au sens très large du terme), comme par exemple la fréquence des mots dans un texte, la couleur dominante d'une image, ... et de déduire de cette observation la catégorie (ou la classe) dans laquelle on doit le ranger. Dans le cas du SPAM, on aura donc deux classes, les courriers normaux et les SPAMS, le filtrage consistant à placer chaque courrier observé dans une des deux classes. Notons que la terminologie "reconnaissance de forme" est plutôt anglo-saxonne (traduction de *pattern recognition*), le terme français le plus courant dans la communauté scientifique étant "discrimination". Je me permets de renvoyer le lecteur intéressé par la discrimination en général à mes cours universitaires sur les réseaux de neurones [5] pour plus d'information.

Évaluation d'un filtrage

Le filtrage de SPAM est donc un cas particulier de discrimination à deux classes. Pour comparer diverses techniques de filtrage, il faut choisir un critère d'évaluation : qu'est-ce qu'un bon filtrage, ou plus généralement, qu'est-ce qu'une bonne discrimination ?

La réponse évidente à cette question est qu'une bonne discrimination doit faire le moins d'erreurs possible ! Cependant, quand on étudie de près la notion d'erreur, la situation se complique. Dans le cas du SPAM, on cherche à obtenir un **taux de détection** élevé : quand un pourriel arrive dans notre boîte, on veut qu'il soit marqué SPAM. Le taux de détection se définit comme le pourcentage de pourriels reçus reconnus comme SPAM (ou encore comme la probabilité que notre filtre dise qu'un courrier reçu est un SPAM, sachant que le courrier est effectivement un SPAM). Mais avoir un taux de détection élevé n'est pas suffisant : il suffit en effet d'utiliser le filtre le plus idiot possible (tout courrier électronique est classé comme SPAM) pour obtenir un taux de 100 %, le filtre se révélant totalement inutile.

Le taux de détection ne mesure en effet que la moitié du problème de discrimination : la qualité du classement d'un SPAM. C'est insuffisant car on reçoit aussi des courriers intéressants (heureusement) qu'il ne faut surtout pas classer comme pourriels. Il faut donc que notre système possède un **taux de fausse alarme** minimal : tout courrier classé comme SPAM doit effectivement en être un. On définit alors le taux de fausse alarme comme le pourcentage de courriers normaux reçus classés comme SPAM (ou encore comme la probabilité que notre filtre dise qu'un courrier reçu est un SPAM, sachant que le courrier n'est pas un SPAM). Le filtre idiot du paragraphe précédent à un taux de fausse alarme de 100 %, ce qui est catastrophique.

De façon générale, un problème de discrimination à deux classes s'évalue toujours en tenant compte des deux taux mentionnés, avec comme but ultime (en général impossible à atteindre) d'avoir 100 % de détection et 0 % de fausse alarme. Il faut noter que des situations très diverses sont possibles, comme par exemple un taux de fausse alarme très faible (par exemple 1 %) associé à un taux de détection moyen (70 %), ou au contraire un taux de détection très satisfaisant (95 %) handicapé par une fausse alarme élevée (10 %). Selon le contexte, on peut privilégier un des deux taux. Dans le cadre du filtrage de SPAM, il est classique de minimiser la fausse alarme même si cela fait légèrement baisser le taux de détection : on préfère lire de temps en temps un SPAM plutôt que de perdre des courriers normaux. Dans d'autres applications, la fausse alarme est autorisée à atteindre un niveau plus élevée car cela permet d'augmenter de façon importante le taux de détection.

Quelques solutions classiques

Le SPAM est une plaie ancienne (mais en progression très rapide) et de nombreuses méthodes de filtrage ont été développées ces dernières années. Voici quelques exemples :

- **les listes noires** : elles constituent très certainement la méthode la plus

ancienne et la plus simple. Il s'agit de constituer une liste permettant d'identifier les *spammeurs*, en se fondant essentiellement sur deux principes :

- certains *spammeurs* respectent la loi (si, si, ça existe) et ne cherchent donc pas à masquer leur identité. Il suffit alors de faire une liste des adresses électroniques de ces *spammeurs* pour filtrer leur courrier ;
- beaucoup de *spammeurs* ne respectent pas la loi et utilisent en particulier des *open relays* pour envoyer leurs messages. Un serveur de *mails en open relay* peut être utilisé par n'importe qui pour envoyer un courrier électronique à n'importe quel destinataire. Il suffit de faire un `telnet` sur le port 25 (`smtp`) du serveur pour profiter de ses services. Encore une fois, une liste des serveurs incriminés permet de filtrer le courrier en provenance de ceux-ci.

Le problème est bien entendu la constitution des listes noires. Il faut en effet analyser les SPAM reçus pour déterminer leurs origines, vérifier que le serveur utilisé est bien un *open relay*, qu'aucun courrier important ne peut venir de ce serveur, etc. La tâche étant lourde et répétitive, la plupart des administrateurs système utilise des listes externes, maintenues par des volontaires ou des entreprises commerciales. Les dérives possibles de ce genre de système sont nombreuses, comme la censure, les erreurs, etc. Je connais même un certain rédacteur en chef (que, par charité, je ne nommerai pas ici) d'une certaine revue de sécurité informatique qui a été victime d'un *blacklistage* intempestif. Paul Graham résume bien la situation dans un de ses articles ([6]) et donne des liens intéressants vers des abus documentés et des évaluations de performances des listes noires, qui souffrent en général d'un taux de détection médiocre doublé d'une fausse alarme élevée !

Fred: à un moment, il y avait même des listes qui ne prenaient pas la peine de vérifier l'état du serveur : on pouvait donc blacklister qui on voulait ...

- **les règles de filtrage** : elles constituent l'idée la plus attrayante pour l'informaticien de formation classique. Le principe est simple : en analysant des SPAMS, on peut extraire des caractéristiques qui les différencient des courriers classiques. Par exemple, un courrier qui contient l'expression *larger penis* est très certainement un SPAM vantant les mérites de pilules magiques. Il suffit donc de mettre au point des règles efficaces qui associent à chaque élément de preuve contenu dans le message un score, puis d'additionner les scores obtenus par le message. Si le score dépasse un certain niveau, on décide que le message est un SPAM. L'exemple le plus connu d'implémentation de telles idées est le logiciel open source *SpamAssassin* [7]. Il s'installe sur un serveur SMTP et ajoute un en-tête aux messages considérés comme du SPAM, ce qui permet ensuite aux utilisateurs de refuser les courriers, de les classer dans une boîte spéciale, etc. Mais ces règles de filtrage ont un gros défaut, l'énorme intervention humaine nécessaire à leur mise en place : il faut lire les SPAMS, identifier les éléments caractéristiques, associer un score à chacun d'eux, vérifier que les interactions d'éléments caractéristiques n'entraînent pas un taux de fausse alarme trop élevé, etc. De plus, les *spammeurs* sont parfaitement informés de l'existence et du contenu des règles de filtrage. Ils modifient donc régulièrement le contenu de leurs envois, en utilisant par exemple des trucs très basiques, comme écrire

Viagra à la place de *Viagra*. L'administrateur système doit donc mettre à jour très régulièrement la base de règles et on se retrouve finalement dans l'affrontement classique entre le glaive et le bouclier.

- **les signatures** : une idée intéressante a été popularisée par le programme *Vipul's Razor* [8], l'exploitation de l'aspect massif des envois de SPAM. Le principe de *Razor* est de fournir une base de données de signatures de SPAM. Les signatures sont construites à partir de messages identifiés par un opérateur humain comme étant du SPAM. Elles sont relativement robustes, c'est-à-dire qu'elles ne sont pas sensibles à de petites modifications (contrairement à une signature MD5 par exemple) tout en permettant une bonne discrimination : un message assez différent de celui utilisé pour construire la signature ne sera pas confondu avec ce dernier.

Quand un message arrive sur un serveur qui utilise *Razor*, la signature de ce dernier est calculée puis envoyée à un serveur central pour être comparée aux signatures stockées dans la base. Le serveur central filtre de cette façon les messages.

Razor repose entièrement sur sa communauté d'utilisateurs. En effet, toute personne peut obtenir un accès au serveur central lui permettant d'envoyer des SPAMS pour qu'ils soient enregistrés dans la base de données. Le serveur maintient une évaluation de la qualité de la signature, qui tient compte de la "notoriété" des personnes qui ont soumis le message correspondant. L'idée est que le SPAM n'est rentable qu'avec l'envoi de millions de messages, ce qui prend du temps. Or, dès qu'un destinataire d'un SPAM est membre de la communauté *Razor*, le message correspondant peut être bloqué par le serveur central. L'approche proposée par *Vipul's Razor* est séduisante, mais elle n'est pas sans défaut, notamment parce que le serveur central est contrôlé par une entreprise commerciale (seul le module client est open source). De plus la qualité du filtrage dépend entièrement de l'algorithme de signature qui est particulièrement délicat à mettre au point. Les *spammeurs* compliquent d'ailleurs la tâche en ajoutant du texte aléatoire dans leurs messages.

L'approche probabiliste

Motivations

La rapide présentation des solutions existantes fait ressortir quelques problèmes :

- une intervention humaine relativement importante est souvent nécessaire ;
- le filtrage dépend des ressources externes au serveur de mail ;
- les performances sont souvent mauvaises.

Il est donc clair que le recours à d'autres solutions n'est pas un luxe. Pourtant, l'utilisation d'un modèle probabiliste ne va pas de soi. Intuitivement en effet, un *mail* est soit un SPAM, soit un courrier normal. Pour le lecteur, le classement est presque instantané et sans aucune ambiguïté. Le problème est que le mécanisme cognitif que nous utilisons pour classer un courrier est extrêmement évolué et s'appuie sur des critères qu'il est très difficile de traduire en algorithme. Nous sommes confrontés ici à

un véritable problème d'intelligence artificielle.

Or, la pratique de la reconnaissance de forme en particulier, et de l'intelligence artificielle en général, a montré que les modèles probabilistes sont très efficaces pour le type de problèmes qui nous concerne, même si cela peut sembler contre-intuitif. La raison profonde de cette efficacité est liée aux capacités limitées de l'ordinateur. Nous utilisons en effet de très nombreuses informations pour dire qu'un courrier est un SPAM. Nous considérons par exemple le titre, l'expéditeur et bien sûr le contenu lui-même. L'ordinateur n'est pas capable de traiter toutes ces informations, en particulier parce qu'elles doivent être combinées. Il considère donc un ensemble limité d'informations. Cette réduction transforme le problème initial déterministe en un problème probabiliste.

Pour comprendre comme cela est possible, considérons un exemple très simple. La plupart des téléphones portables (et certains téléphones fixes) affichent le numéro du correspondant qui tente de vous joindre. Si vous avez une bonne mémoire, ou si le numéro du correspondant est dans le répertoire du téléphone, il est très facile de l'identifier avant de décrocher. Sauf que certains standards téléphoniques remplacent le numéro direct de votre correspondant par celui du standard ! Si vous connaissez plusieurs personnes qui passent par le même standard, vous ne pouvez plus deviner qui vous appelle. Un problème trivial, l'association d'un numéro complet au nom du correspondant, devient brutalement un problème beaucoup plus difficile quand on complique légèrement les données. Où interviennent les probabilités dans cet exemple ? Dans la modélisation du problème, comme nous allons le voir dans les sections suivantes.

Un peu de probabilités

Les probabilités sont en général un domaine mystérieux pour l'informaticien classique, car il est rare que sa formation comporte une présentation avancée (et surtout à but utilitaire) des concepts mis en jeu. Je vais tenter ici de donner quelques exemples simples pour bien montrer sur quoi reposent les outils de filtrage de SPAM les plus efficaces actuellement.

Pour fixer les idées, je note $P(A)$ la probabilité de l'évènement A . On peut interpréter ce nombre de deux façons (je renvoie le lecteur intéressé par l'antagonisme entre les deux visions de la probabilité et, plus généralement, à l'histoire des probabilités, à l'excellent et très accessible livre de Ian Hacking, L'émergence de la probabilité [9]):

- l'approche **fréquentiste** considère qu'on a réalisé de très nombreuses expériences pour lesquelles A peut se produire ou non (par exemple A correspond à "la pièce tombe sur pile" et les expériences sont donc des lancers de cette pièce). La probabilité de A est alors le nombre d'expériences pour lesquelles A a eu lieu divisé par le nombre total d'expériences ;
- l'approche **bayésienne** considère que $P(A)$ représente notre *a priori* sur l'évènement A , notre degré de certitude que A va avoir lieu. Par exemple, nous savons qu'une pièce de monnaie normale a autant de chance de tomber sur pile que sur face, car elle est symétrique. Si A correspond à "la pièce tombe sur pile",

$P(A)$ vaut donc logiquement 0.5.

Reprenons l'exemple de l'identification d'un correspondant proposé dans la section précédente. Si mon téléphone n'est pas capable d'afficher le numéro de ce correspondant, il m'est très difficile de deviner qui m'appelle quand il sonne. Je peux cependant estimer des probabilités en tenant compte du passé. Si, par exemple, sur les cent derniers appels, 30 provenait de ma belle-mère et 20 de ma mère, je peux dire que $P(\text{belle-mère})=3/10$ et que $P(\text{mère})=2/10$, en combinant l'approche fréquentiste (je calcule la probabilité grâce à une fréquence) et l'approche bayésienne (cela me conduit à un *a priori* sur le futur).

Le conditionnement

Si mon téléphone affiche les numéros, mes estimations deviennent beaucoup plus sûres. Par exemple, si je reconnais le numéro de téléphone de mes parents, il n'y a plus vraiment de probabilités, je suis certain qu'il s'agit de mes parents (enfin presque, cela pourrait être quelqu'un de ma famille en visite chez mes parents).

Du point de vue probabiliste, la prise en compte d'évènements dans l'estimation de la probabilité d'un autre évènement s'appelle le conditionnement. C'est un des concepts les plus importants du domaine. On note $P(A|B)$ la probabilité de l'évènement A sachant que l'évènement B a eu lieu. On peut définir cette grandeur par le rapport entre $P(A \text{ et } B)$ et $P(B)$. Intuitivement, il s'agit simplement de compter le nombre de fois que A et B ont eu lieu en même temps et de le diviser par le nombre de fois que B a eu lieu. Je peux de cette façon calculer la probabilité que le coup de fil vienne de mes parents quand leur numéro s'affiche sur mon téléphone, à savoir $P(\text{parents}|\text{numéro parents})$. Si mes parents sont les seuls à m'appeler depuis leur téléphone, alors $P(\text{parents et numéro parents})=P(\text{numéro parents})$, car il y a une relation d'équivalence entre le fait que mes parents appellent et le fait que leur numéro s'affiche. J'obtiens donc ainsi $P(\text{parents}|\text{numéro parents})=1$, il n'y a plus de hasard.

La discrimination probabiliste

Les problèmes de discrimination s'expriment très bien avec le formalisme probabiliste. On cherche en effet à déterminer la classe d'un objet en fonction des connaissances qu'on a de cet objet, ce qui revient à estimer les probabilités $P(\text{objet dans la classe } C|\text{l'objet est décrit par les observations } x)$. Si on sait calculer les probabilités de cette forme (on les appelle les **probabilités a posteriori**), on peut décider de classer l'objet x dans la classe la plus probable. Un algorithme de discrimination qui utilise cette approche est un **classifieur bayésien**. Notons que le terme bayésien est utilisé ici en référence au théorème de Bayes qui permet de faire des calculs sur les probabilités conditionnelles (les $P(A|B)$), et non pas par opposition à l'approche fréquentiste.

En pratique, l'estimation des probabilités est difficile. Reprenons l'exemple du téléphone en le compliquant : nous souhaitons prendre en compte plusieurs évènements apportant chacun de l'information sur un autre évènement. Supposons

que j'observe, par exemple, qu'entre 19h et 20h, 30 % de mes appels (sur mon téléphone fixe et sur mon téléphone portable) proviennent de ma mère. Supposons de plus qu'en général 10 % des appels que je reçois sur mon téléphone portable proviennent de ma mère (indépendamment de l'heure de l'appel). Il est 19h30 et mon téléphone portable sonne. Quelle est la probabilité que l'appel vienne de ma mère ? Sans information additionnelle, on ne peut pas répondre à cette question. C'est pourquoi on fait parfois une hypothèse très forte qui consiste à dire que **les caractéristiques observées sont conditionnellement indépendantes relativement à la classe de l'objet**, du charabia pour le non probabiliste ! Avant de donner plus d'explication, notons qu'un classifieur qui fait cette hypothèse est appelé un **classifieur bayésien naïf**.

Précisons donc cette propriété. On dit que deux événements sont indépendants si $P(A \text{ et } B) = P(A)P(B)$. En pratique cela signifie que l'occurrence de l'évènement B ne donne aucune information sur celle de l'évènement A (car $P(A|B) = P(A)$ quand A et B sont indépendants et que $P(B) > 0$). On dit que deux événements sont conditionnellement indépendants relativement à un troisième si $P(A \text{ et } B|C) = P(A|C)P(B|C)$. L'hypothèse évoquée au-dessus correspond donc au fait que recevoir un appel entre 19h et 20h et recevoir un appel sur le téléphone portable sont conditionnellement indépendants relativement à l'évènement "l'appel provient de ma mère", mais aussi relativement à l'évènement "l'appel ne provient pas de ma mère" (il y a ici deux classes, les appels de ma mère et les autres).

Comment utiliser cette hypothèse en pratique ? Les calculs sont assez lourds, c'est pourquoi je ne les inclus pas ici, mais ils conduisent à une propriété remarquable. On obtient en effet que $P(\text{mère} | 19h30 \text{ et portable})$ est donné par $ab / (ab + (1-a)(1-b)p / (1-p))$ avec $a = P(\text{mère} | \text{portable})$, $b = P(\text{mère} | 19h \text{ à } 20h)$, $p = P(\text{mère})$. Or, nous connaissons toutes ces valeurs par hypothèse ($a = 0.1$, $b = 0.3$ et $p = 0.2$) et nous obtenons $P(\text{mère} | 19h30 \text{ et portable}) = 0.16$. Le point vraiment important est qu'avec l'hypothèse d'indépendance conditionnelle, on peut toujours calculer les probabilités à posteriori grâce à des probabilités faciles à calculer. En fait, on calcule $P(\text{objet dans la classe } C | \text{l'objet est décrit par les observations } x_1, x_2, \dots, x_n)$ en fonction de :

- les $P(C)$ pour toutes les classes : le calcul est facile, il suffit de compter le nombre total d'objets dans chaque classe puis de diviser chaque effectif par le nombre total d'objets.
- les $P(x_1|C)$, $P(x_2|C)$, etc. Le calcul est aussi relativement facile : pour chaque classe, il suffit de compter le nombre d'objets qui exhibent la caractéristique x_1 , puis de diviser ce nombre par l'effectif de la classe.

Bien entendu, pour compter les objets, encore faut-il connaître leur classe. Ce problème se règle toujours de la même manière en discrimination, même si on n'utilise pas un classifieur naïf : on emploie un ensemble dit **d'apprentissage**. C'est une collection d'objets classés correctement et sensés être représentatifs du problème étudié.

Dans la réalité, l'approche dite bayésienne naïve fonctionne rarement. Il est en effet difficile d'isoler les caractéristiques importantes des objets pour faire les comptages, l'hypothèse d'indépendance conditionnelle est souvent beaucoup trop forte, etc.

Cependant, dans le cas du filtrage de SPAM, les performances obtenues sont remarquables ! De plus, le volume des données rend difficile l'utilisation de techniques beaucoup plus sophistiquées.

Application au SPAM

Katia: je pense que la section suivante sur l'article de Graham pourrait faire un bon encadré

L'article de Paul Graham

L'article de Paul Graham, *A plan for Spam* [3] a eu un énorme succès et peut être considéré comme l'origine de l'intérêt renouvelé pour le filtrage de SPAM par méthodes statistiques. C'est à la fois un bien, car il est clair que la visibilité de ces méthodes et leur implémentation dans des logiciels grand public (comme dans l'outil de *mail* de Mozilla depuis la version 1.3 [10]) sont en grande partie dues à cet article. Cependant, l'article de Graham a une renommée et une visibilité largement imméritées pour diverses raisons :

- l'algorithme proposé, bien que fonctionnel, est formellement faux : il utilise une règle de combinaison de probabilités fautive (Graham oublie de tenir compte des $P(C)$ pour les classes étudiées) ;
- aucun travail antérieur n'est cité alors que Graham est très loin d'être l'inventeur du filtrage bayésien naïf (cf par exemple [11] et [12], cités par l'article suivant de Graham [13]);
- l'article fait une confusion entre les classifieurs bayésiens en général et l'approche bayésienne naïve sur laquelle est construit l'algorithme.

On pourrait continuer la liste assez longtemps, ce qui ne serait pas très charitable : Paul Graham est un spécialiste du lisp, pas un statisticien. De plus, il a corrigé en partie ses erreurs dans un complément à son premier article, intitulé *Better Bayesian Filtering* [13]. Enfin, son algorithme fonctionne en pratique !

Principes de base

Le filtrage de SPAM est un problème de discrimination à deux classes, on souhaite donc estimer $P(SPAM|contenu\ du\ mail)$. Comme nous l'avons vu dans la section précédente, l'hypothèse d'indépendance conditionnelle simplifie le calcul de cette probabilité en la ramenant aux calculs de $P(SPAM)$ (probabilité de la classe SPAM) et des $P(caractéristique\ du\ mail|SPAM)$ pour les caractéristiques retenues. Nous avons donc à régler deux problèmes :

1. choisir les caractéristiques intéressantes dans les *mails*
2. les utiliser pour estimer les probabilités

Comme nous l'avons dit plus haut, le second problème se règle grâce à un ensemble d'apprentissage. Dans notre cas particulier, il s'agit donc d'un ensemble de courriers électroniques, contenant à la fois des SPAM et des *mails* normaux. **Point crucial, les**

courriers doivent être correctement étiquetés et être représentatifs du mail reçu, à la fois pour les SPAMS et pour les courriers normaux.

Comme nous reconnaissons un SPAM grâce à son contenu, il est logique de chercher dans celui-ci des caractéristiques intéressantes pour le filtrage. Pour estimer des probabilités de façon satisfaisante, il faut que chaque caractéristique retenue apparaisse dans un nombre minimal de *mails*. De plus, il faut que l'extraction des caractéristiques soit rapide. C'est pourquoi on se focalise sur les "mots" contenus dans les messages. Plus précisément, on extrait de l'intégralité de chaque message (*header* inclus) des *tokens* correspondant aux mots du message, mais aussi aux commandes javascript, aux balises HTML, etc. Ces *tokens* sont les caractéristiques du message (sensées être indépendantes...) dont les probabilités sont estimées par comptage. On transforme ainsi l'ensemble d'apprentissage en une énorme table de hachage qui à un *token T* associe la probabilité $P(T|SPAM)$. On estime aussi $P(SPAM)$ et $P(non\ SPAM)=1-P(SPAM)$.

Quand un nouveau message arrive, il est découpé en *tokens*, auxquels on associe les probabilités $P(T|SPAM)$. En utilisant les formules induites par l'hypothèse d'indépendance conditionnelle, on calcule $P(SPAM|message)$ par combinaison des probabilités retrouvées dans la table de hachage. Si la probabilité obtenue est élevée, on considère le message comme du SPAM.

Détails techniques

Dans la pratique, les principes de base énoncés plus haut ne sont pas suffisants. On rencontre par exemple les problèmes suivants :

- Comment définir les *tokens* ?
- Doit-on garder tous les *tokens* de l'ensemble d'apprentissage ? Non, les probabilités estimées pour les *tokens* très peu fréquents sont très imprécises ce qui fausse l'estimation globale. Il faut donc un seuil d'apparition pour être pris en compte (Graham propose d'exiger cinq occurrences d'un *token* pour son intégration dans les calculs) ;
- Comment traiter les *tokens* des messages entrants qui n'apparaissent pas dans la table de hachage ? Graham propose de leur affecter une probabilité $P(T|SPAM)$ de 0.4, de façon totalement heuristique ;
- Doit-on prendre en compte tous les *tokens* d'un message entrant pour calculer sa probabilité d'être du SPAM ? Graham propose une règle heuristique : ne conserver que les 15 *tokens* les plus significatifs, c'est-à-dire ceux pour lesquels $P(T|SPAM)$ est le plus éloigné possible de 0.5 (soit vers 0, soit vers 1).
- A partir de quelle valeur de $P(SPAM|message)$ doit-on marquer le message comme du SPAM ? Graham propose la valeur de 0.9

Les solutions proposées par Graham sont très heuristiques et relativement peu satisfaisantes du point de vue mathématique. Il obtient pourtant des résultats excellents, avec un taux de détection de 99.5 % et une fausse alarme de 0.03 % (sur ses propres données, il ne faut pas en déduire qu'on aura toujours d'aussi bons résultats). Pour beaucoup de statisticiens, ces bons résultats montrent simplement

que le filtrage de SPAM est un problème très simple statistiquement. Certains outils évolués de filtrage probabiliste utilisent d'ailleurs un support théorique beaucoup plus évolué que les heuristiques de Graham et obtiennent de meilleurs résultats. On peut citer par exemple `spambayes` (cf [14] et [15]) et `bogofilter` (basé sur les mêmes idées que `spambayes`, cf [16] et [17] pour des évaluations de performances). L'ancêtre du filtrage de *mails* avec un classifieur bayésien naïf, `ifile`, qui date de 1996, repose lui aussi sur beaucoup moins d'heuristiques que l'algorithme de Graham (cf [18]).

Performances et évolutivité

Divers travaux expérimentaux ont été menés par les groupes qui implémentent des classifieurs probabilistes de SPAM. Les performances sont en général excellentes, avec un taux de détection au dessus de 98 % et un taux de fausse alarme presque nul (cf [17] par exemple). Pour tout praticien de la reconnaissance de forme, ces valeurs sont tout simplement extraordinaires tant il est rare d'obtenir un tel succès sur un problème réel. Du point de vue pratique, elles se traduisent par un grand confort pour l'utilisateur. Le site de nouvelles linux `lwn.net` a fait une petite expérience [19] avec deux semaines de messages envoyés à son adresse principale (`lwn@lwn.net`), ce qui représente 3000 *emails* dont 2705 de SPAM (!). Après apprentissage, `bogofilter` a un taux de fausse alarme de 0 et un taux de détection de 96.5 %, c'est-à-dire que 94 SPAM sur 2705 ne sont pas détectés. Le nombre de SPAM qui arrivent effectivement à l'utilisateur devient ainsi inférieur au nombre de messages normaux. Ces chiffres ne sont qu'une expérience parmi d'autres et sont optimistes (quand on évalue les performances d'un classifieur sur l'ensemble d'apprentissage, les résultats sont toujours meilleurs que sur de nouvelles données), mais ils sont révélateurs d'une tendance : le filtrage probabiliste fonctionne, contrairement aux anciennes méthodes.

De plus, le filtrage est très facilement évolutif. On peut en effet livrer chaque logiciel avec des informations résumant l'ensemble d'apprentissage organisées de sorte à ce qu'une mise à jour de la table de hachage soit simple. L'utilisateur est alors à même de modifier le comportement du filtre pour l'adapter à ses besoins. A chaque fois qu'il reçoit un nouveau SPAM, l'utilisateur l'ajoute (s'il le souhaite) à la base d'apprentissage. Le logiciel recalcule les probabilités associées aux *tokens* et le filtre évolue en fonction du profil des messages reçus par l'utilisateur (cette stratégie est mise en oeuvre par mozilla [10]). L'intervention humaine est minimale alors que les efforts que doivent consentir les *spammeurs* pour contourner le filtre sont énormes et sûrement vains (cf [3], [6] et [13] par exemple). On peut même mettre en place des adresses pièges qui transmettent directement les messages qu'elles reçoivent dans l'ensemble des SPAMS utilisés pour l'apprentissage.

En pratique

Le filtrage de SPAM par méthodes probabilistes est implémenté par de très nombreuses solutions open source. J'ai déjà indiqué mozilla [10] au niveau client, `bogofilter` [16] et `spambayes` [14] au niveau serveur. D'autres implémentations existent, comme `crm114` [20] qui s'utilise au niveau serveur ou client, ou encore `spamassassin` [7] dont les dernières versions combinent base de règles et méthodes probabilistes. Les idées que j'ai présentées dans cet article sont en fait très populaires, comme l'atteste

la liste conséquente de filtres probabilistes maintenue par Paul Graham [21]. L'évaluation comparative de ces solutions dépasse cependant largement le cadre de cet article.

Conclusion

Peut-on gagner la guerre contre le SPAM ? Pendant de nombreuses années, le modèle commercial du SPAM a semblé presque sans faille, tant les efforts consentis pour lutter contre les *spammeurs* étaient importants, avec la clé de très maigres résultats. Depuis un peu moins d'un an, la situation a radicalement évolué, grâce à l'article de Paul Graham, *A plan for Spam*. Graham a réussi à populariser des méthodes anciennes, simples à implémenter et extrêmement efficaces, les classifieurs bayésiens naïfs. En quelques mois, les implémentations libres ont fleuri. Peu à peu, les utilisateurs adoptent les outils de filtrage probabilistes inspirés de l'algorithme de Graham. Il semblerait bien que pour la première fois depuis son apparition, le SPAM perde des points. Pour l'instant, le volume de pourriels ne baisse pas, mais la gêne pour l'internaute diminue radicalement, tant le filtrage est efficace : on passe facilement d'une centaine de SPAMS par jour à quelques uns. L'efficacité du filtrage laisse espérer une baisse du taux de retour des SPAMS et, à terme, une diminution radicale de l'intérêt de ces *mailings* massifs. Au niveau de l'utilisateur, le filtrage probabiliste du SPAM laisse entrevoir un monde meilleur. Les mois à venir diront si les nouvelles méthodes tiennent leurs promesses !

Références

- [1] Citée par Cnet news.com : <http://news.com.com/2100-1023-979108.html>
- [2] Étude de ePrivacy Group : <http://www.eprivacygroup.com/article/articlestatic/58/1/6>
- [3] Paul Graham - *A plan for Spam*. Août 2002 (<http://www.paulgraham.com/paulgraham/spam.html>)
- [4] Graffiti : <http://www.palm.com/products/input/>
- [5] Cours sur les réseaux de neurones : <http://apiacoa.org/teaching/nn/>
- [6] Paul Graham - *Filters vs. Blacklists*. Septembre 2002 (<http://www.paulgraham.com/paulgraham/falsepositives.html>)
- [7] <http://www.spamassassin.org>
- [8] <http://razor.sourceforge.net/>
- [9] Ian Hacking - *L'émergence de la probabilité*. 2002. Liber (Seuil).
- [10] <http://www.mozilla.org/mailnews/spam.html>
- [11] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz - *A Bayesian Approach to Filtering Junk E-mail* Proceedings of AAAI-98 Workshop on Learning for Text Categorization. Disponible en ligne à : <http://research.microsoft.com/~horvitz/junkfilter.htm>
- [12] Patrick Pantel and Dekang Lin. - *SpamCop - A Spam Classification & Organization Program*. Proceedings of AAAI-98 Workshop on Learning for Text Categorization. Disponible en ligne à : <http://www.cs.ualberta.ca/~ppantel/Content/papers.htm>
- [13] Paul Graham - *Better Bayesian Filtering*. Janvier 2003 (

<http://www.paulgraham.com/paulgraham/better.html>)

- [14] <http://spambayes.sourceforge.net/>
 - [15] Un article de Gary Robinson qui explique certains principes à la base de spambayes : <http://radio.weblogs.com/0101454/stories/2002/09/16/spamDetection.html>
 - [16] <http://bogofilter.sourceforge.net/>
 - [17] Évaluations de performances du filtrage de bogofilter par Greg Louis : <http://www.bgl.nu/bogofilter/>
 - [18] <http://www.nongnu.org/iframe/>
 - [19] Expérience de lwn : <http://lwn.net/Articles/9460/>
 - [20] <http://crm114.sourceforge.net/>
 - [21] <http://www.paulgraham.com/paulgraham/filters.html>
-

Fabrice Rossi
Université Paris-IX Dauphine
CEREMADE
Fabrice.Rossi@apiacoa.org
<http://apiacoa.org>