Filtre antispam par analyse statistique bayésienne (**)

Sujet proposé par Didier Rémy

`Didier.Remy[at]inria.fr`

Page de suivi à l'adresse :
`http://pauillac.inria.fr/remy/poly/bayesian/suivi.html`

Résumé

Le projet consiste à réaliser un programme qui permet de filtrer les spams parmi les courriers arrivant en utilisant une analyse statistique bayésienne. Le programme utilise deux corpus, l'un constitué de spams et l'autre constitué de «bons» courriers, qui pourront être augmentés incrémentalement, permettant ainsi une forme d'apprentissage. Dans un premier temps, le programme digère les deux corpus pour fabriquer une base de données attribuant des probabilités à certains événements élémentaires. La base de données est ensuite utilisée pour calculer à l'arrivée d'un nouveau courrier la probabilité pour que celui-ci soit un spam.

Principe

Pour estimer la probabilité qu'un courrier soit un spam, on se ramène à observer des événements élémentaires, typiquement la présence d'un mot donné dans le message, mais on pourra considérer des événements élémentaires plus complexes comme la présence d'un groupe de mots voisins. En fait, on peut s'abstraire de la notion d'événement élémentaire dans la plus grande partie de l'analyse.

La première étape revient à calculer la probabilité p_e pour qu'un courrier soit un spam, sachant que l'événement élémentaire e apparaît dans celui-ci. On calcule la probabilité p_e empiriquement en considérant le rapport des fréquences q_e et $\bar{q}_e$ d'apparition de l'événement e dans le corpus des spams et dans le corpus des «bons» courriers, respectivement. Ainsi, $p_e/\bar{p}_e = q_e/\bar{q}_e$ où $\bar{p}_e$ est la probabilité complémentaire $1 - p_e$. D'où $p = q_e/(q_e + \bar{q}_e)$. On peut ainsi compiler les corpus en une base de donnée $e \mapsto p_e$ qui permet de retrouver rapidement les valeurs de cette fonction par interrogation en un point sans avoir à reparcourir les corpus.

Le deuxième étape consiste à calculer la probabilité qu'un nouveau courrier soit un spam sachant qu'il contient plusieurs événements élémentaires pour chacun desquels on connaît la probabilité d'apparaître dans un spam. À l'arrivée d'un nouveau courrier m, on peut en déduire, sachant qu'il contient un événement élémentaire e, la probabilité p_m que ce soit un spam. En fait, on peut obtenir une probabilité plus précise en observant en parallèle plusieurs événements élémentaires. En pratique, on observe parmi tous ceux qui apparaissant dans le courrier m les k événements $e_i^{i\in[1..k]}$ les plus significatifs $e_i^{i\in[1..k]}$, c'est-à-dire ceux dont la probabilité est la plus éloignée de la médiane 0.5. En supposant les événements élémentaires indépendants, ce qui est évidemment faux, mais donne empiriquement de très bons résultats, on a $p_m = P_m/(P_m + \bar{P}_m)$ où $P_m = \Pi_{i=1}^k p_{e_i}$ et $\bar{P}_m = \Pi_{i=1}^k \bar{p}_{e_i}$ (voir [3] pour plus de détails).

Il ne reste plus qu'à considérer une limite haute ℓ et une limite basse $\bar{\ell}$ à partir desquels on peut considérer que le courrier est un spam ou au contraire un bon courrier. Entre les deux, on est incertain. Cependant, on va en général traiter les courriers incertains comme bons afin de minimiser les faux positifs (classer comme «spams» des courriers qui n'en sont pas) qui sont nettement plus graves que les faux négatifs (classer comme bons courriers des spams).

Un des intérêts de ce système est qu'il permet un apprentissage incrémental : lorsqu'un courrier a été mal filtré, c'est-à-dire qu'un spam se retrouve dans les bons courriers ou inversement, il suffit d'ajouter le courrier au corpus correspondant et de mettre à jour la base de donnée, ce qui peut se faire de façon incrémentale. Aussi le système *apprend* chaque fois que ses erreurs lui sont signalées.

Quelques ajustements

En pratique [1], on triche un peu pour le calcul des fréquences. D'une part, cela permet d'éviter les cas dégénérés où un mot n'apparaît pas assez souvent pour être significatif dans un des corpus—en effet une fréquence ne doit pas être nulle ni infinie. D'autre part, cela introduit des biais volontaires qui favorisent les faux négatifs plutôt que les faux positifs. .

Aussi, on choisira dans un premier temps la formule suivante, très biaisée, mais dont l'efficacité à été vérifiée en pratique [1] :

$$p = \begin{cases} 0.5 & \text{si } N_e + \bar{N}_e < 5 \\ \min\left(p_{max},\ \max\left(p_{min},\ \left(\dfrac{q_e}{q_e + \bar{q}_e}\right)\right)\right) & \text{sinon} \end{cases}$$

avec $q_e = \min 1 (\ (cN_e/N))$ (et symétriquement pour $\bar{q}_e$) où c est une pondération que l'on pourra ajuster, N_e est le nombre total d'occurrences de e comptées avec leur multiplicité (donc le nombre total de fois que e est apparu dans un message quelconque) et N le nombre de messages dans le corpus des spams (et symétriquement pour $\bar{c}$, $\bar{N}_e$ et $\bar{N}$). Cette formule corrige les effets liés à la petite taille de l'échantillon (un mot n'apparaît dans aucun des deux corpus). D'autre part, elle prend en compte la multiplicité des occurrences de façon très *ad hoc*, ce qu'elle corrige en bornant la fréquence à 1. Enfin, elle donne un poids différent aux mots apparaissant dans chacun des corpus.

Dans [1], on choisit $c = 1$ et $\bar{c} = 2$ pour biaiser le caclul des fréquences et les probabilités des événements élémentaires son bornées avec $p_{min} = 0.01$ et $p_{max} = 0.99$. On choisit également $\ell = 0.95$ et $\bar{\ell} = 0.05$ et $k = 15$.

Variations possibles

Par défaut, un événement correspond à l'occurrence d'un mot dans le message (en-tête ou corps). On pourra considérer d'autres formes d'événements, par exemple en distinguant les occurrences dans l'en-tête des occurrences dans le corps du message, ou bien en choisissant des paires de mots consécutifs comme événements élémentaires [2].

Le découpage d'un courrier en mots doit être réglé correctement. En fait, le découpage simple qui consiste à identifier les majuscules et minuscules, ajouter les chiffres, le signe $ et les apostrophes aux lettres et considérer tous les autres caractères comme des séparateurs donne expérimentalement de bons résultats [1]. Bien sûr, d'autres découpages sont possibles et pourront être explorés.

Enfin, on pourra revenir sur les approximations faites pour le calcul des probabilités élémentaires ou leurs combinaisons. (Toutefois, il ne faudra pas être surpris si un modèle plus fidèle à la théorie donne parfois de moins bons résultats en pratique.)

Mise en oeuvre

Dans un premier temps on pourra traiter le message brut (en-tête comprise) sans l'analyser et ne traiter que les messages simples sans attachements. On pourra, *en extension*, reconnaître les attachements dans les corps de messages, ce qui permet de passer à l'échelle : on ignorera alors tous les attachements sauf ceux de type `text` ou `html`.

On initialisera le programme avec des corpus de courriers de taille raisonnable (au moins de l'ordre du millier d'entrées).

Comme le mode interrogation est fait pour être appelé fréquemment, on veillera à ce qu'il soit raisonnablement efficace. Ce qui implique de maintenir la base de donnée (typiquement une table de hachage) dans un format suffisamment compact.

Interface

Le programme devra pouvoir être appelé de deux façons différentes, d'une part pour ajouter des messages dans le corpus, d'autre part pour marquer des nouveaux courriers. On utilisera *impérativement* l'interface suivante :

```
programme db add ( -spam | -good | mailbox )*
programme db mark mailbox*
```

où *`programme`* est le nom donné au programme. Le premier argument *`db`* est le nom de la base de donnée des corpus «digérés». Si la base de donnée désignée par *`db`* n'existe pas, elle est créée vide. Le second argument indique le mode (`add` pour l'ajout dans le corpus de nouveaux courriers et `mark` pour l'interrogation).

Dans le cas d'un ajout, chacun des arguments restants est soit un des deux drapeaux `-spam` ou `-good` indiquant comment traiter les boîtes aux lettres qui suivent, soit un fichier représentant une boîte aux lettres *`mailbox`*. On peut ainsi alterner les fichiers de spams et les fichiers de bons courriers.

Dans le cas de l'interrogation, les arguments restants sont des boîtes aux lettres à analyser. Si aucune boîte aux lettres n'est spéficiée alors le programme considèrera son entrée standard comme une boîte aux lettres. La commande consiste alors pour chaque boîte aux lettres à analyser chacun des messages et recopier le message après avoir ajouté à la fin de l'en-tête un champ supplémentaire `X-Spam: yes;` *`p`* `;` *`s`* où p est la probabilité d'être un spam sous forme d'un nombre flottant à deux chiffres, compris entre `0.00` et `1.00`. On pourra aussi indiquer à la fin du champ `X-Spam` la liste *`s`* des événements les plus significatifs avec leur probabilité d'indiquer un spam, par exemple, sous la forme d'une suite d'éléments *`e`*`:`*`p`* séparés par des espaces. Les messages sont ensuite ajoutés les uns à la suite des autres pour former une boîte aux lettres imprimée dans la sortie standard. Les boîtes aux lettres en entrée et en sortie sont toutes supposées au format `mbox` d'Unix, brièvement décrit ci-dessous.

Format des boîtes aux lettres

Le format `mbox` d'Unix permet de représenter une séquence de courriers dans un fichier, appelé boîte aux lettres. Une boîte aux lettres est une suite de messages séparés par une ligne vide, *i.e.* deux caractères `LF` (code ASCII 10) qui se suivent.

Un message commence une ligne (*i.e.* commence le fichier ou suit un caractère `LF`) par exactement la chaîne `From` suivi d'un espace. Le reste de la ligne donne l'adresse de l'expéditeur et la date d'envoi. Ensuite vient, après la fin de la première ligne, le message proprement dit. Un message est composé d'un en-tête suivi d'un corps optionnel, qui est dans ce cas séparé de l'en-tête par une ligne vide.

Pour plus de détails, on pourra se reporter au manuel Unix de mbox (ou faire `man mbox`) pour le format des boîtes aux lettres, à la RFC 822 pour le format des messages et à la RFC 2045 pour le format des attachements dans les corps des messages.

Quelques Références

[1] Graham 2002, A Plan for Spam, 2002 (`http://www.paulgraham.com/spam.html`).

[2] Graham 2003, Better Bayesian Filtering 2003 (`http://www.paulgraham.com//better.html`).

[3] Combining probabilities (`http://www.mathpages.com/home/kmath267.htm`).

[4] Abelard, Cause, Chance and Bayesian statistics (`http://www.abelard.org/briefings/bayes.htm`).

[5] RFC 822 (`http://www.faqs.org/rfcs/rfc822.html`)

[6] RFC 2045 (`http://www.faqs.org/rfcs/rfc2045.html`)

[7] Manuel unix du format mbox (`http://www.cebitec.uni-bielefeld.de/cgi-bin/man.cgi?section=5&topic=mbox`)