Contribution au classement statistique mutualisé de messages électroniques (spam)

José Márcio Martins da Cruz

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

MINES ParisTech

École doctorale n°432 : Sciences des Métiers de l'Ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure des Mines de Paris

Spécialité
« Informatique Temps-Réel, Robotique et Automatique »

présentée et soutenue publiquement par

José Márcio MARTINS DA CRUZ

le 13 octobre 2011

Contribution au classement statistique mutualisé de messages électroniques *(spam)*.

Directeur de thèse : **Alain GALLI**

Jury

Alexis Nasr, Professeur, LIF, Université Aix-Marseille 2, Marseille	Président
Patrick Gallinari, Professeur, LIP 6, Université Pierre et Marie Curie, Paris	Rapporteur
Mihai Mitrea, Maître Assistant, ARTEMIS, Institut Télécom SudParis, Évry	Rapporteur
Eric Allman, Chief Science Officer, Sendmail Inc., Oakland, CA, U.S.A	Examinateur
Gladys Huberman, Professeur, CCSI, Mines-Paristech, Paris	Examinateur
Alain Galli, Directeur de Recherche, CERNA, Mines-Paristech, Paris	Directeur de Thèse

T H È S E

MINES ParisTech
Centre de Robotique
60, bd Saint-Michel, 75272 - Paris CEDEX, France

École doctorale n°432 : Sciences des Métiers de l'Ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure des Mines de Paris

Spécialité
« Informatique temps-réel, Robotique et Automatique »

présentée et soutenue publiquement par

José Márcio MARTINS DA CRUZ

le 13 octobre 2011

Contribution au classement statistique mutualisé de messages électroniques *(spam)*.

Directeur de thèse : **Alain GALLI**

Jury

Alexis Nasr, Professeur, LIF, Université Aix-Marseille 2, Marseille	Président
Patrick Gallinari, Professeur, LIP 6, Université Pierre et Marie Curie, Paris	Rapporteur
Mihai Mitrea, Maître Assistant, ARTEMIS, Institut Télécom SudParis, Évry	Rapporteur
Eric Allman, Chief Science Officer, Sendmail Inc., Oakland, CA, U.S.A	Examinateur
Gladys Huberman, Professeur, CCSI, Mines-Paristech, Paris	Examinateur
Alain Galli, Directeur de Recherche, CERNA, Mines-Paristech, Paris	Directeur de Thèse

MINES ParisTech
Centre de Robotique
60, bd Saint-Michel, 75272 - Paris CEDEX, France

À mes parents

Marcelino da Cruz
(1899 - 2000)

Palmira Pereira Martins da Cruz
(1912 - 2002)

Remerciements

Cette thèse est l'aboutissement d'un rêve resté en attente pendant plus de trente ans. Avec beaucoup de plaisir, j'adresse quelques remerciements à ceux qui de près ou de loin ont contribué à ce qu'il devienne réalité et à ceux qui peuplent ma pensée et mes bons souvenirs.

Tout d'abord mes parents, *Palmira* et *Marcelino*, qui ne sont plus mais qui ont participé aux premiers pas de ce rêve. Ils m'ont appris qu'il faut rêver et persévérer pendant toute la vie et que les réussites, pour les avoir, il faut les mériter. Ils m'ont appris qu'il faut toujours aller plus loin. C'est à eux que je dédie ce rêve qu'ils n'ont pas pu voir l'aboutissement.

Je remercie vivement les membres du jury qui ont accepté de donner leur avis et juger mon travail. Les rapporteurs, par la lecture minutieuse et les remarques pertinentes, m'ont permis d'améliorer le contenu. Je remercie particulièrement *Eric Allman*, sans qui la messagerie électronique n'existerait pas ou ne serait pas ce qu'elle est aujourd'hui.

Alain Galli, mon directeur de thèse, m'a beaucoup aidé avec des idées et suggestions, et pas seulement dans le domaine statistique. Il a supporté mon entêtement : encadrer quelqu'un qui a ses propres idées n'est pas une tâche facile.

Les discussions avec *Gordon V. Cormack*, professeur à l'Université de Waterloo, m'ont été très instructives : des idées et suggestions pertinentes m'ont guidé dans certains points de cette thèse. J'ai eu l'honneur de faire deux publications avec lui.

Ce sujet de thèse n'a pas reçu de financement et a été traité avec les moyens disponibles. La messagerie de l'*École des Mines de Paris* utilise les résultats de mes travaux depuis presque dix ans et m'a permis de collecter des données et de valider les résultats. Je tiens particulièrement à remercier *Gladys Huberman*, directrice du *Centre de Calcul et Systèmes d'Information* pour le soutien apporté pendant toutes ces années. Je tiens à remercier M. *Michel Schmitt*, Directeur de la Recherche, pour le soutien et les discussions que j'ai eu plaisir d'avoir avec lui. Bien entendu, tous mes collègues de service, pour les discussions, suggestions et la patience avec mon "fort caractère".

Cinq *"cobayes"* m'ont donné accès à leurs boîtes aux lettres légitimes. Sans eux, une partie importante de cette thèse n'aurait pas pu être réalisée : *Thierry Weil*, *Alain Galli*, *Chakib Bouallou*, *Jean-Michel Viovy* et *Michel Gaudet*.

Un nom inoubliable est *Serge Aumont*, du *Comité Réseau des Universités*. Dans un domaine où le nombre de experts est aussi important que le nombre de gestionnaires de messagerie, il m'a toujours soutenu et encouragé. C'est devenu un vrai ami.

N'oublions pas les utilisateurs du logiciel libre *j-chkmail*, avec des suggestions, des rapports de bugs, des retours d'expérience, de l'amitié et même des donations.

Et puis, il y a des rencontres qui ont marqué mon existence. Ils n'ont pas contribué directement à cette thèse mais j'aimerais profiter de l'occasion pour leur rendre hommage et leur dire que je ne les oublie pas : *Lys et Gilmar Ferreira*, *Maria Laura et Carlos Anybal Pilles Patto*, *Zélia et Antonio Adalberto dos Santos*, *Jean-Claude Babault* et *Claude Lepeinteur*.

Dans un passé encore plus lointain, les enseignants que j'ai eu au collège et au lycée ont contribué à réveiller mon intérêt par les disciplines scientifiques. J'ai une pensée spéciale pour *Père Luis Marconetti*, mon professeur de chimie au lycée (Colégio Dom Bosco), il y a quarante

ans. Tant que ma mémoire me le permet, il est celui qui m'a appris à regarder les disciplines scientifiques avec des modèles et de la rigueur, il m'a transmis l'envie d'aller plus loin que ce qui était enseigné en classe.

Il y a aussi quelqu'un très spécial : ma soeur *Lucenne* ! *Lucenne* m'a fait cadeau du préface, m'a encouragé tout au long de ce rêve et a représente, virtuellement, mes parents dans l'aboutissement de cette aventure. C'est *la soeur* avec qui je peux tout partager : nos *"histoires et blagues internes"* : des choses que seuls des frères sont capables de comprendre, et de se comprendre... C'est ma complice depuis mon enfance.

Et finalement ma famille : des nombreuses heures ont été sacrifiées pour que je puisse réaliser ce projet. Ma famille a compris ô combien ce rêve était important pour moi et m'a soutenu. J'espère pouvoir transmettre à mes enfants cette envie d'acquisition de connaissances et d'aller toujours plus loin, c'est le plus grand héritage que j'ai eu de mon père.

Abstract

Since the 90's, different machine learning methods were investigated and applied to the email classification problem (*spam filtering*), with very good but not perfect results. It was always considered that these methods are well adapted to filter messages to a single user and not filter to messages of a large set of users, like a community.

Our approach was, at first, look for a better understanding of handled data, with the help of a corpus of real messages, before studying new algorithms. With the help of a simple linear discriminator classifier with online active learning, we could show empirically that with a simple classification algorithm coupled with a learning strategy well adapted to the real context it's possible to get results which are as good as those we can get with more complex algorithms. We also show, empirically, with the help of messages from a small group of users, that the efficiency loss is not very high when the classifier is shared by a group of users.

Résumé

Depuis la fin des années 90, les différentes méthodes issues de l'apprentissage artificiel ont été étudiées et appliquées au problème de classement de messages électroniques (*filtrage de spam*), avec des résultats très bons, mais pas parfaits. Il a toujours été considéré que ces méthodes étaient adaptées aux solutions de filtrage orientées vers un seul destinataire et non pas au classement des messages d'une communauté entière.

Dans cette thèse notre démarche a été, d'abord, de chercher à mieux comprendre les caractéristiques des données manipulées, à l'aide de corpus réels de messages, avant de proposer des nouveaux algorithmes. Puis, nous avons utilisé un simple classificateur discriminant linéaire avec de l'apprentissage actif en ligne - pour démontrer empiriquement qu'avec un algorithme simple et une configuration d'apprentissage adaptée au contexte réel de classement, on peut obtenir des résultats aussi bons que ceux que l'on obtient avec des algorithmes plus complexes. Nous avons aussi démontré, avec des ensembles de messages d'un petit groupe d'utilisateurs, que la perte d'efficacité peut ne pas être significative dans un contexte de classement mutualisé.

Préface

Nous sommes frères, José-Márcio et moi. Nous sommes nés dans des terres trempées, lesquelles terres ouvrent violemment leurs portes aux eaux de la rivière Paraguay et forment ainsi des dessins transparents cachés par le teint bleu des eaux et vert des forêts : le Pantanal du Brésil. Nous sommes venus au monde à travers des parents qui nous ont appris qu'avoir de la joie et du courage signifie avoir de la santé pendant toute la vie. Ils ont été notre port. Nous avons été élevés proches du sol, nous passions notre temps à faire du calcul mental, sans comparaison. Maintenant, je retrouve ces numéros, tel un solo d'abstractions, dans la thèse de mon frère le plus jeune.

Dans le tronc de la langue portugaise il existe un mot qui m'affecte par le côté gauche du courage et peut-être à cause de la quantité de sel que je porte dans les entrailles de mon corps, résultat d'une chose qui s'hérite même dans l'utérus, le piédestal sacré et la maison d'habitation. Ce mot est *Finisterra*. Le sens quasi entier de ce mot nous est donné par la géographie physique et par le sentiment lusitanien de se lancer à la mer. De la géographie vient la notion de point le plus occidental du continent européen, un terroir de pierre entouré par la mer et par le ciel. Le sentiment vient de l'homme qu'habite ce terroir et qui, à l'époque des Découvertes, a pensé à l'horizon comme la ligne qui ferme son territoire. *Finisterra* était, pour les portugais, cette pointe de terre à l'extremité vers le sud-ouest de Portugal où un belvédère imaginaire est transpercé : l'horizon[1]. Comme héritage et honneur paternel nous avons reçu un fragment intact de ce mot *Finisterra* pour nos vies. Cela est notre communion de biens.

José-Márcio accompli sa thèse, il marche avec aplomb, ici la terre finit, le bout le plus au sud de la terre héritée, qu'un jour va se séparer, va être en contact avec la mer et faire croire que la fin de la terre est le premier acte. Il est impossible de rester à la frontière, il est dans la limite. Par limite, je comprends une grandeur constante, de qui une autre grandeur peut s'approcher indéfiniment sans jamais l'atteindre. Il y a toujours des pas à faire même si la marque zéro est récrée, rétablie.

Mon coeur se réchauffe de savoir que, par le passé, quand nous faisions du vélo sur la terre rouge des rues de notre petit village campagnard, il n'y avait pas une destinée déjà tracée, mais une histoire à écrire de sa propre main, comme cette thèse.

Et libertés !

Lucenne Cruz
Rio de Janeiro, Brésil, le 12 juin 2011

[1] N.d.T. - Au Portugal, *Finisterra* fait référence au *Cabo da Roca* sis à 42 km à l'ouest de Lisbonne. Le poète Luis de Camões (1525-1580) décrivait le cap dans les *Lusiadas* comme «l'endroit où la terre s'arrête et où la mer commence». Source Wikipédia.

Prefácio

Somos irmãos, José-Márcio e eu. Nascemos em terras encharcadas, aquelas terras que escancaram suas portas para as águas do rio Paraguai formando assim desenhos transparentes cobertos de tintas azuis das águas e verdes das matas : o Pantanal do Brasil. Viemos ao mundo através de pais que nos ensinaram que a alegria e a coragem é saúde para a vida toda. Eles são o nosso porto. A gente foi criada rente ao chão, ganhava o tempo cuidando de fazer contas de cabeça, sem comparações. Vejo de volta esses números, um solo de abstrações, na tese do meu irmão mais novo.

Há no tronco da língua portuguesa, uma palavra que me afeta pelo lado esquerdo da coragem e talvez pela quantidade de sal que carrego nos vãos do meu corpo por conta de uma coisa que se herda ainda no útero, o andor sagrado e a casa de habitação. Essa palavra é *Finisterra*. O sentido quase inteiro dessa palavra nos é dado pela geografia física e pelo sentimento lusitano de se lançar ao mar. Da geografia, vem a noção de ponto mais ocidental do continente europeu que é um chão de pedra cercado de mar e céu. O outro vem do homem que habita esse solo e, que na época dos Descobrimentos, pensou o horizonte como a linha que fecha o seu território. *Finisterra* era, para os portugueses, aquela ponta de terra no extremo sudoeste de Portugal onde se fincou um mirante imaginário : o horizonte. Por herança e honra paterna, recebemos um fragmento inteiro dessa palavra *Finisterra* para as nossas vidas. Essa é a nossa comunhão de bens.

José-Márcio finda a sua tese, aqui se acaba a terra, o cabo mais ao sul daquela herdade que se anda firme, mas que chega um dia que há-de se romper àquela ponta de terra e alcançar o mar e fazer crer que o fim da terra é o primeiro ato. Não tem como ficar na fronteira, está-se no limite. Por limite, entendo uma grandeza constante, da qual outra grandeza pode aproximar-se indefinida sem nunca a atingir. Há sempre passos a dar ainda que à estaca zero, que se recria, que se recomeça.

Aquece o meu coração saber que, lá atrás quando andávamos de bicicleta pelas ruas de terra vermelha da nossa cidade do interior, não havia designo traçado, mas sim uma história a ser escrita pelo próprio punho, como esta tese.

E liberdades !

Lucenne Cruz
Rio de Janeiro, Brasil, 12 de junho de 2011

Table des matières

Remerciements **iii**

Abstract **v**

Résumé **vii**

Préface **ix**

Prefácio **xi**

I Introduction **1**

1 Introduction **3**
1.1 Introduction . . . 3
1.2 Un *spam*, un *ham*, c'est quoi, exactement ? . . . 4
1.3 Le filtrage de spam basé sur le contenu . . . 5
1.4 De quoi parle-t-on dans cette thèse... . . . 5
1.5 Travaux similaires et contributions de cette thèse . . . 7
1.6 Organisation de ce document . . . 8

2 L'environnement d'un filtre anti-spam **9**
2.1 Introduction . . . 9
2.2 Anatomie d'un message électronique . . . 9
2.3 Le processus de filtrage . . . 9
2.3.1 Aspects Temporels . . . 12
2.3.2 Interactions avec le destinataire . . . 13
2.3.3 Interactions avec l'expéditeur . . . 13
2.4 L'apprentissage . . . 14
2.5 Le filtre . . . 14
2.6 La représentation des messages . . . 15

3 Historique **17**
3.1 Introduction . . . 17
3.2 La communauté de la recherche . . . 17
3.2.1 Les débuts : Expérimentations avec classificateurs . . . 17
3.2.2 Le filtrage de spam : un problème à part entière . . . 19

3.2.3 La contribution de TREC - Spam Track . . . 20
3.3 Les praticiens et développeurs de logiciels libres . . . 22
3.3.1 Les classificateurs à règles fixes . . . 22
3.3.2 Les classificateurs adaptatifs (avec apprentissage) . . . 22
3.4 Discussion, Controverses et Conclusions . . . 23

II Les briques d'un classificateur de messages électroniques 25

4 La Représentation des Messages Électroniques 27
4.1 Introduction . . . 27
4.2 Génération et Représentation des Messages . . . 28
4.2.1 Génération des Messages . . . 28
4.2.2 Représentation . . . 28
4.2.3 Vocabulaire et Dictionnaire . . . 29
4.2.4 Boîtes aux lettres . . . 29
4.3 Segmentation du texte : approche linguistique . . . 30
4.3.1 Niveau mots (ou formes simples) . . . 30
4.3.2 Niveau formes multiples . . . 31
4.3.3 Niveau sub-mots . . . 32
4.3.4 Niveaux sémantique et pragmatique . . . 33
4.4 Méta-attributs et termes synthétiques . . . 33
4.5 Contenu textuel ou méta-informations ? . . . 33
4.6 Sélection des attributs - réduction de la dimension . . . 36
4.6.1 Sélection de sous-ensembles d'attributs . . . 36
4.6.2 Extraction (ou construction) d'attributs . . . 38
4.6.3 Réduction de dimension dans les applications de filtrage de spam . . . 39
4.7 Normalisation de la taille des documents . . . 39
4.8 La Multiplicité des Langues . . . 40
4.9 Conclusions . . . 42

5 L'Apprentissage Artificiel 43
5.1 Introduction . . . 43
5.2 Apprentissage Statistique . . . 44
5.3 Séparabilité . . . 46
5.3.1 La notion de séparabilité . . . 46
5.3.2 Des sous-ensembles séparables par un classificateur . . . 47
5.3.3 Sous-ensembles linéairement séparables . . . 47
5.4 Les modes d'apprentissage . . . 48
5.4.1 Apprentissage supervisé et non supervisé . . . 48
5.4.2 Apprentissage hors ligne (batch) versus en ligne . . . 48
5.4.3 Apprentissage actif et apprentissage passif . . . 49
5.5 Conclusions . . . 51

6 Les Algorithmes de Classement 53
6.1 Introduction . . . 53
6.2 Le classificateur Bayésien . . . 53
6.2.1 Modèles événementiels . . . 54
6.2.2 Apprentissage et implémentation . . . 56
6.2.3 Classificateur Linéaire . . . 56
6.2.4 Les classificateurs bayésien naïfs et les logiciels libres . . . 56
6.2.5 Discussion et Conclusions . . . 57
6.3 Le *Perceptron* . . . 57
6.4 Régression Logistique . . . 58

6.5 Machines à Vecteur de Support - (SVM) 59
6.6 Notes bibliographiques 59

III Mutualisation du classement de messages électroniques 61

7 L'utilisation mutualisée d'un filtre anti-spam 63
7.1 Introduction 63
7.2 La taxonomie des communautés 64
7.3 L'interaction avec les destinataires 64
7.4 La taxonomie des solutions de filtrage mutualisé 66
7.5 Les modes d'apprentissage courants 68
7.6 Discussion et Conclusions 68

8 Caractéristiques spatiotemporelles d'un flot de messages 71
8.1 Introduction 71
8.2 Décalage et dérive 71
8.2.1 La dérive temporelle 73
8.2.2 Le décalage spatial 75
8.3 La diversité 75
8.4 L'apprentissage dans un flot non stationnaire 75
8.4.1 Apprentissage sur les exemples pris dans une fenêtre temporelle 76
8.4.2 Apprentissage incrémental 76
8.4.3 L'apprentissage en *mini-batches* 77
8.5 Conclusions 78

9 Un filtre anti-spam avec apprentissage actif en ligne 79
9.1 Classement et apprentissage actif en ligne 79
9.2 Représentation des messages 80
9.3 SLDC - Simple Linear Discriminative Classifier 81
9.3.1 Apprentissage 81
9.3.2 Apprentissage Actif 82

IV Expérimentations 83

10 Caractéristiques temporelles empiriques des flots de messages 85
10.1 Introduction 85
10.2 L'évolution de la répartition des messages par classe 85
10.3 L'impact de l'âge et de l'âge relatif des exemples 87
10.4 Les caractéristiques temporelles d'un flot de messages 88
10.4.1 Introduction 88
10.4.2 Les données brutes 90
10.4.3 La tendance à long terme (dérive linéaire) des séries 91
10.4.4 Les demi-variogrammes des séries 94
10.4.5 Caractérisation par ajustement d'un modèle dynamique linéaire 94
10.4.6 Analyse Spectrale 99
10.4.7 Comparaison avec autres classificateurs 103
10.5 Conclusions 107

11 Expérimentations de classement avec apprentissage actif en ligne **109**
11.1 Introduction . . . 109
11.2 Objectifs et Hypothèses . . . 109
11.3 Modèle fonctionnel et contexte de simulation . . . 110
11.3.1 Les modules du simulateur . . . 110
11.3.2 Corpus de messages . . . 112
11.3.3 Protocole de simulation . . . 113
11.4 Résultats . . . 114
11.4.1 Les différents flots de messages . . . 114
11.4.2 Le flot *JM-H/JM-S* . . . 121
11.5 Expérimentation de classement mutualisé . . . 126
11.6 Conclusions . . . 126

V Réflexions à approfondir **131**

12 Geométrie des classes et filtrage parfait **133**
12.1 Introduction . . . 133
12.2 Séparabilité et Classement sans Erreur . . . 133
12.3 Distribution spatiale des données . . . 134
12.3.1 Apprentissage de fonctions booléennes . . . 134
12.3.2 Discussion . . . 137
12.4 Vocabulaires des classes : communs ou disjoints ? . . . 137
12.5 Discussion et Conclusions . . . 140

13 Comparaison de flots ou ensembles de messages **143**
13.1 Introduction . . . 143
13.2 Les approches pour comparer des ensembles de messages . . . 144
13.2.1 Les boîtes noires . . . 144
13.2.2 Les classificateurs génératifs . . . 144
13.2.3 Les classificateurs discriminants . . . 144
13.3 Le divergences entre distributions de probabilité . . . 145
13.3.1 Divergence de Kullback-Leibler . . . 146
13.3.2 Divergence de Jensen-Shannon . . . 146
13.3.3 Les *f-divergences* . . . 147
13.3.4 Le cosinus et la distance euclidienne . . . 148
13.4 Propriétés des Divergences . . . 148
13.4.1 Le lemme de Neymann-Pearson . . . 149
13.4.2 Lemme de Chernoff-Stein . . . 150
13.5 Résultats Préliminaires . . . 151
13.6 Discussion et Conclusions . . . 151
13.7 Notes Historiques et Bibliographiques . . . 153

14 Modèles de Mélange Fini **155**
14.1 Introduction . . . 155
14.2 Modèles de mélange et commuté . . . 156
14.3 Discussion et perspectives . . . 156

VI Conclusions **159**

15 Conclusions **161**
15.1 Résultats . . . 161
15.2 Perspectives . . . 162

15.2.1 Déploiement 162
15.2.2 Les améliorations de l'algorithme d'apprentissage 163
15.2.3 Chapitres "réflexions à approfondir" 163
15.2.4 Validation plus large 163
15.3 Que faut-il retenir ? 163

VII Annexes **165**

A Probabilités et Statistique **167**
A.1 Définitions 167
A.2 Moments d'une distribution de probabilités 167
A.3 Estimation de probabilité 168
A.4 Modélisation Statistique et Critères d'Information 168
A.5 Notes Bibliographiques 169

B Processus Stochastiques et Séries Temporelles **171**
B.1 Stationnarité 171
B.2 Outils 172
B.2.1 Corrélogramme d'une Série Temporelle 172
B.2.2 Variogramme 172
B.2.3 Periodogramme 172
B.3 Modèles de Séries Temporelles 173
B.3.1 Modèles Linéaires 173
B.3.2 Modèles Non Stationnaires 174
B.3.3 Modèles avec Saisonnalité 174
B.3.4 Ajustement de Modèles 174
B.4 Notes Bibliographiques 175

C Approximation stochastique **177**

D Métriques d'évaluation d'un classificateur de messages électroniques **179**
D.1 Tables de contingence et indicateurs dérivés 179
D.2 *ROC* (Receiver Operating Characteristic) 180
D.3 Notes Bibliographiques 181

E Théorie d'Information **183**
E.1 Définitions 183

F Treillis et Algèbre de Boole **185**
F.1 Ensembles 185
F.2 Ordre Partiel et Treillis 186
F.3 Algèbre de Boole 187

G Résultats détaillés - Chapitre 11 **189**

H Publications **209**

I Distinctions et Prix **211**
I.1 Sendmail - 25 Years of Trusted Messaging 211
I.2 Terena Networking Conference 2005 - Selected Papers 212

J Le logiciel de filtrage de spam *j-chkmail* **213**

K Glossaire **215**

Bibliographie **230**

Première partie

Introduction

CHAPITRE 1

Introduction

Il faut se méfier des ingénieurs. Ça commence par la machine à coudre et ça finit par la bombe atomique.

Marcel Pagnol

1.1 Introduction

Internet a commencé à exister dans les années 70, à l'initiative du DARPA, pour faciliter les communications entre les chercheurs et les départements de la défense américains. Dans les années 90, Internet est devenu un outil grand public et est, petit à petit, pris une part importante dans le fonctionnement de notre société, aussi bien sur le plan individuel que dans les entreprises et organismes de l'administration, grâce aux applications Web et à la messagerie électronique. Dans les organisations professionnelles, ces applications ont pris une place si importante que leur activité parfois cesse en cas d'indisponibilité ou mauvais fonctionnement.

(a) La viande en boîte produite par Hormel Foods

(b) Une scène du sketch de Monty Python

FIG. 1.1: L'utilisation du mot *Spam* pour désigner les messages indésirables a été inspiré par un sketch de *Monty-Python* sur un produit de *Hormel Foods* à base de viande épicée vendu dans des boîtes

Parmi les plaies se propageant par la messagerie électronique se trouvent les *virus* et les *spams*. Ces derniers sont les messages électroniques envoyés en masse et aveuglement proposant,

par exemple, toute une panoplie de produits pharmaceutiques, de la contrefaçon ou encore de la pornographie.

Jusqu'à la fin des années 90 l'activité *spam* était restée marginale. Aujourd'hui, les estimations divergent mais globalement on estime entre 70 % et 95 % la fraction du trafic *SMTP* sur Internet résultant de cette seule activité. Il s'agit d'une gêne au trafic, puisqu'il faut dimensionner les infrastructures réseau en conséquence, et aussi une perte de temps pour les destinataires de ces messages.

Depuis la fin des années 1990, une panoplie de solutions de filtrage sont apparues, certaines basées sur l'identification du chemin parcouru par le message et d'autres basées sur le contenu des messages, certaines certaines objectives et d'autres moins voire même farfelues, avec propagation d'un certain nombre de mythes.

Un de ces mythes consiste à dire que l'utilisation mutualisé d'un filtre de contenu pour classer les messages d'une communauté n'est pas faisable puisque *"les boites aux lettres de personnes différentes sont différentes"*. Le but de cette thèse est justement l'étude de ce contexte de filtrage et démontrer que cela est possible avec des solutions relativement simples.

1.2 Un *spam*, un *ham*, c'est quoi, exactement ?

SPAM® [1] est un produit à base de viande épicée et conditionné en boîte. Ce mot est formé des initiales de "Shoulder Pork and hAM"/"SPiced hAM". En 1970, le groupe *Monty Python Flying Circus* a présenté un sketch qui se passait dans un restaurant où le SPAM® entrait dans la composition de tous les plats du menu. Le sketch finissait par une cacophonie où tous chantaient : *"Spam, spam, spam, spam, spam, spam, spam, spam, lovely spam ! Wonderful spam !"*[2]. Le caractère répétitif et non souhaité du mot spam dans ce sketch a conduit la communauté d'internet à l'utiliser pour se référer à cette catégorie de messages indésirables et répétitifs.

Il y a plusieurs définitions de *spam*, certaines plus restrictives que d'autres. Les entreprises de marketing direct, par exemple, essaient de promouvoir une définition assez faible de façon à ce que de la publicité, même sauvage, ne soit pas considérée comme du *spam*.

Dans le contexte de cette thèse, nous avons considéré comme *spam* les messages satisfaisant, en même temps, les trois critères suivants :

1. les messages n'ont pas été sollicités et n'ont aucun intérêt ;
2. les messages ont été envoyés en masse ;
3. le destinataire ne connait pas l'expéditeur (même si l'inverse peut ne pas être vrai).

Cependant, cette définition admet une appréciation subjective, en particulier du premier critère, où l'expression *"non sollicité"* est remplacé par *"non souhaité"*. Ce flou est néanmoins inévitable et ajoute une incertitude dans les résultats, que nous avons pu observer dans la partie expérimentale.

Malgré la diversité des définitions et les controverses, cette définition semble être la plus acceptable car elle tend vers ce que le destinataire est généralement prêt à accepter. C'est, à notre avis, l'objectif de toute application de classement : la satisfaction de son utilisateur.

Les messages indésirables sont parfois aussi désignés par *UBE* (Unsollicited Bulk Email) ou *UCE* (Unsollicited Commercial Email).

Les messages légitimes, par opposition aux *spams*, sont souvent désignés par le mot *ham* - probablement pour dire que "*spam*, c'est mauvais mais *ham*, c'est bon".

[1] SPAM® est une marque déposée de Hormel Foods - `http://www.spam.com`

[2] On peut retrouver ce sketch sur internet, par exemple, à `http://www.youtube.com/watch?v=0DshB09FQ8w` ou `http://www.montypython.net/scripts/spamskit.php`

1.3 Le filtrage de spam basé sur le contenu

Une approche souvent utilisée pour filtrer le *spam* est de vérifier si un message en cours d'examen satisfait un certain nombre de critères - des règles. Si oui, le message est refusé et, dans le cas contraire, le message est accepté. Ces critères sont par exemple : la présence de l'expéditeur dans une liste noire, un nombre important de messages du même expéditeur dans une période très courte ou encore la présence de certains mots (*viagra*, *pornographie*, ...) dans le contenu du message. En général, un seul critère ne suffit pas pour atteindre un niveau d'efficacité satisfaisant : il faut alors les combiner. Mais la combinaison optimale n'est pas forcément triviale à trouver et, assez souvent, cela se fait grâce à des simplifications pas toujours justifiées.

Dans une approche naïve, les critères sont établis à l'avance et manuellement. La démarche consiste à définir une fonction ayant comme domaine les messages à classer (ceux déjà vus mais aussi ceux à venir), représentés par les valeurs prises par les critères, et comme image, les deux classes possibles : *ham* et *spam*. La difficulté résulte du fait que, à notre connaissance, il n'y pas de modèle mathématique permettant d'associer une classe à un message et, s'il y en avait un, il serait très complexe.

L'approche alternative, l'*apprentissage artificiel*, consiste à utiliser un ensemble d'exemples, chacun avec son étiquette de classement, de taille suffisante pour être représentatif de l'ensemble des messages à classer, et de laisser un *"algorithme" "apprendre"* la relation fonctionnelle existant entre chaque exemple et sa classe associée et être capable de généraliser cette relation à des cas non vus dans les exemples.

Un ensemble de critères est toujours nécessaire. Soit les critères sont renseignés explicitement, soit on se contente de définir une heuristique permettant à *"l'algorithme"* de construire lui-même cet ensemble de critères. C'est ce qui se passe dans une application de classement d'objets textuels, où le dictionnaire est construit pendant l'apprentissage et non pas établi à l'avance. Dans les applications d'apprentissage artificiel, ces critères sont appelés *"attributs"* (*features*) et peuvent correspondre, par exemple, à la présence ou absence d'un mot du dictionnaire dans le message.

L'approche par apprentissage artificiel est particulièrement intéressante lorsqu'il n'y a pas de modèle mathématique ou alors il y en a un mais il est trop complexe. Cette approche, appliquée au contenu des messages, est celle qui nous intéresse dans cette thèse, même si nous reconnaissons que ce n'est pas la seule approche efficace.

Il existe une dualité entre l'*apprentissage artificiel* et l'*inférence statistique* [260, p. 11], avec utilisation de termes différents pour représenter les mêmes choses. Mais il y a une différence de principe entre ces deux domaines : le premier s'intéresse plus à l'aspect algorithmique du problème tandis que le deuxième se préoccupe de la compréhension et de la modélisation des données. L'auteur de cette thèse, plus versé dans la partie informatique, estime, et ce n'est que son avis personnel, que la technique est arrivée à un point où l'amélioration de l'efficacité des filtres actuels passe par une meilleure compréhension des données.

1.4 De quoi parle-t-on dans cette thèse...

On parle, bien sûr, de filtrage de *spam* ! Mais, par qui, pour qui et pourquoi ?

A l'origine, l'auteur a traité ce sujet dans un cadre de gestion d'un service de messagerie et développement d'un logiciel libre de filtrage.

Les techniques employées actuellement dans les logiciels libres ou commerciaux semblent avoir atteint leurs limites. La recherche de nouvelles méthodes de classement de messages est devenue indispensable. Les grands fournisseurs de solutions de filtrage tablent sur des méthodes telles que les listes noires[3] ou les listes de réputation[4]. L'expérience montre que ces solutions permettent de dégrossir largement le flot de messages, mais lorsque l'on cherche une efficacité plus importante, il faut faire appel à des méthodes de filtrage basées sur le contenu, en particulier des classificateurs statistiques.

[3]Par exemple, Spamhaus - `http://www.spamhaus.org`

[4]Par exemple, Cisco/Ironport `http://www.senderbase.org`

L'utilisation des classificateurs statistiques a souvent été considérée d'un intérêt limité à un usage individuel. Les résultats de recherche traitant de leur utilisation partagée sont, à notre connaissance, très rares et ne permettent pas de tirer des conclusions.

Cette thèse étudie l'utilisation partagée d'un classificateur statistique dans une communauté telle qu'une université ou un organisme de recherche, avec des milliers d'utilisateurs de la messagerie, des centres d'intérêt assez diversifiés, mais qui ont quand même quelques points communs.

L'objectif n'est pas de proposer une solution définitive à la problématique du spam. Après tant d'expérimentations faites avec toute sorte de méthode de classement, il nous a semblé utile de faire une pause et de chercher une meilleure compréhension de la problématique du spam.

Depuis une dizaine d'années, l'application de classificateurs statistiques au filtrage de spam est dominée par les classificateurs "dit bayésiens" développés par les praticiens des logiciels libres. La recherche s'intéresse à la problématique du *spam* depuis longtemps, avant même les praticiens mais, comme nous le verrons dans le chapitre consacré à l'historique, il y a un fossé considérable entre ces deux communautés qui ont, parfois, du mal à se parler. Cette thèse essaie de combler cette lacune par une modeste "incursion" dans les domaines d'apprentissage artificiel et statistique, sans pour autant prétendre être une thèse dans ces spécialités.

La démarche de cette thèse est, en partie, inspirée par le abstract d'un article publié par David Hand :

> A great many tools have been developed for supervised classification, ranging from early methods such as linear discriminant analysis through to modern developments such as neural networks and support vector machines. A large number of comparative studies have been conducted in attempts to establish the relative superiority of these methods. [...] these comparisons often fail to take into account important aspects of real problems, so that the apparent superiority of more sophisticated methods may be something of an illusion. In particular, simple methods typically yield performance almost as good as more sophisticated methods, to the extent that the difference in performance may be swamped by other sources of uncertainty that generally are not considered in the classical supervised classification paradigm.
>
> *David Hand - Classifier Technology and the Illusion of Progress* [125]

Cette remarque faite à la marge d'un article de conférence ainsi qu'une autre de Leo Breiman [37], faite quelques années auparavant, ont suscité des réactions et commentaires intéressants de chercheurs reconnus tels que D. R. Cox, Brad Efron, Emanuel Parzen, confirmant la pertinence de la remarque.

David Hand défend l'idée que de nombreux travaux de recherche en classificateurs automatiques et complexes, se font sans tenir compte des conditions réelles de fonctionnement et que parfois des outils simples suffiraient pour obtenir la même efficacité que des outils plus complexes. Leo Breiman compare deux cultures : celle des statisticiens et celle des spécialistes de la modélisation algorithmique (intelligence artificielle), avec un penchant pour cette dernière. Il ressort de ce dialogue que ces démarches sont toutes complémentaires et nécessaires.

Ces remarques ont été faites dans des contextes de classement autres que celui du classement de messages électroniques, avec une portée générale sur la problématique de classement utilisant des techniques d'apprentissage artificiel.

Ceci explique notre démarche. Des nombreux travaux ont été publiés sur le spam mais, à notre connaissance et humble avis, assez peu ont été vraiment évalués dans des conditions réelles et se sont limité à l'aspect algorithmique du problème.

Notre démarche a consisté à :

- utiliser un classificateur relativement simple, adapté au contexte réel. Nous avons choisi un classificateur discriminant linéaire avec apprentissage actif en ligne ;
- utiliser des données réelles, collectés sur une période assez longue ;
- comprendre, le mieux possible, les limitations liées au contexte ;
- comparer les résultats obtenus avec des données réels et synthétiques, et publiés par ailleurs.

1.5 Travaux similaires et contributions de cette thèse

Des nombreux travaux portant sur des points particuliers de la problématique de filtrage de spam ont été publiés, notamment des propositions d'algorithmes et de méthodes de filtrage. Les travaux qui nous ont semblé les plus intéressants sont ceux de Gordon Cormack (*e.g.* [60], [69], [57] ou [59]), qui a été le premier à chercher à sortir de la logique de recherche du "meilleur algorithme" de classement de spam. Comme nous verrons dans le chapitre sur le historique, sa contribution principale porte, d'une part, sur la constatation que le filtrage de messages électroniques est un problème de classement en ligne et non en batch et, d'autre part, sur l'élaboration d'une méthodologie d'évaluation de filtres, basée sur l'utilisation d'un corpus unique et commun de messages.

La première contribution de cette thèse porte sur l'amélioration de la connaissance de la problématique du *spam*. Quasiment tous les algorithmes connus en apprentissage artificiel ont déjà été expérimentés, avec des résultats très bons. La question qui se pose est : faut-il chercher des algorithmes encore plus performants ou faut-il étudier les données pour comprendre ce qui peut empêcher d'aller plus loin ? Nous avons choisi la deuxième option. Pour cela, nous utilisons des données réelles et non plus synthétiques, et nous étudions l'évolution temporelle et le résultat de classement de flots de messages à l'aide d'outils tels les séries temporelles et des demi-variogrammes.

La deuxième contribution résulte de notre démarche donne suite au commentaire de *Hand* discuté dans la Section 1.4. Nous avons cherché à utiliser un algorithme de classement aussi simple que possible, mais défini "astucieusement", de façon à pouvoir identifier ses possibles faiblesses. L'identification de ces faiblesses permettra, par la suite, de rechercher des solutions plus performantes.

Assez souvent, on considère que les caractéristiques des flots de messages varient beaucoup avec leur âge et qu'un classificateur doit impérativement être construit avec des messages de même âge que les messages à classer. Avec Gordon Cormack [71] [81] nous avons démontré que si l'on prend la précaution de supprimer les références temporelles, la dérive des caractéristiques statistiques des messages n'est pas aussi importante, ce qui nous permet d'utiliser, dans certaines limites, indifféremment des messages récents ou plus anciens dans l'apprentissage d'un filtre.

La contribution qui ne relève pas d'une amélioration de la connaissance, dans le cas particulier du filtrage de spam, a été la combinaison d'une boucle de retour d'information de classement correct et l'apprentissage en ligne par approximation stochastique.

L'objectif initial de cette thèse était l'étude de l'utilisation partagée d'un filtre de messages, basé sur le contenu, dans une communauté. Ce but ne peut être envisagé que si les ensembles de messages des différents membres de la communauté ont un minimum de ressemblance. Nous avons utilisé le classificateur simple pour démontrer que dans les conditions d'expérimentation l'efficacité restait très bonne aussi bien dans le cas où les messages étaient destinés à un seul utilisateur ou à un petit groupe, assez hétérogène, pour qui nous avons pu collecter des échantillons de messages.

Dans une deuxième partie de nos contributions (ou plutôt des perspectives), nous avons effleuré quelques domaines permettant de mieux connaitre et/ou modéliser les flots de messages. Ces voies n'ont pas été complètement traitées, mais nous avons estimé utile de dédier une partie finale où nous les mentionnons dans leur état d'avancement. Ce sont des voies que nous envisageons d'y travailler par la suite.

Il est souvent admis dans les communications scientifiques, sans aucune précision ni qualitative ni quantitative, que les messages reçus par des destinataires distincts ne se ressemblent pas. Pour valider cette hypothèse, nous avons eu besoin de comparer des ensembles de messages. Il n'était pas question, bien sûr, de comparer les messages caractère par caractère, mais leur représentation. Ce dont nous avons besoin est d'être capables d'ordonner des ensembles de messages selon leur écart vu par les classificateurs. Nous proposons de représenter des ensembles de messages par la distribution empirique de probabilité des termes trouvés dans les messages et d'utiliser les mesures usuelles de similarité entre distributions pour évaluer la ressemblance entre

les ensembles. Des exemples de critères sont la divergence de Kullback-Leibler, Jensen-Shannon et autres distances/divergences similaires. Cette démarche est probablement valable (encore à démontrer) pour des classificateurs génératifs tels le Bayésien Naïf, où le critère de classement peut être vu comme une comparaison de divergences de Kullback-Leibler.

Nous avons aussi effleuré les modèles de mélange de distributions. En effet, chacun d'entre nous a déjà l'habitude de classer les messages par thème selon certains points communs (*e.g.* thème, groupe de travail, ...). Cela fait que ces messages ont déjà des ressemblances. L'ensemble des messages d'un utilisateur résulte d'une combinaison linéaire de dossiers. A un niveau plus élevé, *e.g.* un service, l'ensemble des messages de tous les utilisateurs du service est aussi une combinaison linéaire des messages de chaque utilisateur. Cette hiérarchie se prête bien à une modélisation par un modèle de mélange, probablement plus dans une optique d'amélioration des connaissances que dans un contexte de conception d'un système de classement de messages.

Un dernier point effleuré est une étude de la "géométrie" de la distribution spatiale des exemples : où nous essayons de représenter les ensembles des messages par un treillis algébrique et essayons de les situer et d'analyser la séparabilité des classes par rapport à, par exemple, les attributs communs et disjoints dans chaque classe.

1.6 Organisation de ce document

Cette thèse est organisée en six parties.

La première partie constitue une introduction : une brève présentation de l'environnement de classement de messages électroniques suivi de l'historique du classement statistique basé sur le contenu.

La partie suivante contient trois chapitres qui présentent brièvement les briques logiques d'un filtre anti-spam : la représentation des messages, les algorithmes de classement et l'apprentissage. Dans chaque chapitre nous mettons en valeur ce qui est relevant pour le problème de filtrage de spam.

La troisième partie décrit le problème qui nous concerne : le classement mutualisé de messages, basé sur le contenu. Dans cette partie, nous examinons les problèmes qui apparaissent ou qui prennent de l'importance dans le contexte de filtrage mutualisé de spam. Comme conséquence, nous proposons une architecture de classement, la plus simple possible pour tenir compte de la démarche choisie (cf section 1.4), mais adaptée aux problèmes soulevés dans cette partie.

La partie suivante présente des résultats expérimentaux obtenus. La première partie des expérimentations visent obtenir une meilleure connaissance des caractéristiques temporelles d'un flot de messages. La deuxième partie, présente les résultats obtenus avec le classificateur simple sur les messages d'un seul destinataire, avec ensembles de messages synthétiques et réels, avec classement mutualisé ou pas.

La partie *"Réflexions à approfondir"* contient des points que nous n'avons qu'effleuré ou qui n'ont été qu'en partie, mais qui constituent des pistes de réflexion nous semblant utiles à approfondir. Ces réflexions portent sur trois aspects : la représentation spatiale des messages et la facilité ou difficulté de classement, des possibles méthodes de comparaison de flots ou ensembles de messages et enfin, la possibilité de représenter de façon hiérarchique un ensemble de messages (*c.à.d.* des modèles de mélange).

Enfin, un chapitre dédié aux conclusions et des annexes.

CHAPITRE 2

L'environnement d'un filtre anti-spam

Un bon croquis vaut mieux qu'un long discours.

Napoléon Bonaparte

2.1 Introduction

Ce chapitre propose une vue globale des filtres anti-spam : l'environnement et les parties constituantes. La plupart des concepts présentés ici seront approfondis, individuellement, dans les chapitres suivants.

2.2 Anatomie d'un message électronique

La Figure 2.2 présente le découpage d'un message électronique, avec son découpage ainsi que les parties de protocole d'échange entre deux dispositifs client et serveur de messagerie.

À noter que les adresses de messagerie que l'on voit dans l'enveloppe ne correspondent pas à celles des en-têtes : le routage du message est effectué selon les adresses de l'enveloppe - une situation possible aussi dans le routage des courriers papier traditionnels.

2.3 Le processus de filtrage

Malgré les nombreux scénarios possibles, le modèle logique d'un processus de filtrage de spam est assez simple. La Figure 2.3 présente un de ces scénarios.

Les messages sont soumis au filtre, dans l'ordre de leur arrivée[1]. Après traitement, le filtre associe le message à une des classes - *ham* ou *spam* - indiquant, éventuellement, l'incertitude du classement à l'aide d'une valeur numérique (score). Le destinataire (un être humain) reçoit le message, valide ou rectifie le classement proposé par le filtre : les messages utiles sont pris en compte et les spams supprimés. Dans un autre scénario, le filtre peut mettre dans un sas (quarantaine) les *spams probables*. En tout cas, le destinataire doit pouvoir corriger les erreurs de classement.

[1]Pour être précis, les filtres, placés sur une passerelle de messagerie, peuvent recevoir et traiter simultanément plusieurs messages, mais les messages sont toujours mis, un par un, dans la boîte aux lettres du destinataire

```
Trying 194.214.158.200...
Connected to paris.ensmp.fr.
Escape character is '^]'.
<--- 220 paris.ensmp.fr ESMTP Sendmail 8.14.4/8.14.4
---> HELO saci.ensmp.fr
<--- 250 paris.ensmp.fr Hello saci, pleased to meet you
---> MAIL from:<alice@ensmp.fr>
<--- 250 2.1.0 <alice@ensmp.fr>... Sender ok
---> RCPT to:<bob@ensmp.fr>
<--- 250 2.1.5 <bob@ensmp.fr>... Recipient ok
---> RCPT to:<charlie@ensmp.fr>
<--- 250 2.1.5 <charlie@ensmp.fr>... Recipient ok
---> DATA
<--- 354 Enter mail, end with "." on a line by itself

                    En-têtes
  From: Jose-Marcio Martins <jose@ensmp.fr>
  To: Jean-Claude Dupont <jean-claude@ensmp.fr>
  Subject: Un message de test
  Date: Sun, 12 Dec 2010 19:32:42 -0200

                  Corps du message
  Salut Jean-Claude,

  Comment vas-tu ? Ceci est juste un message de test !

  Joe

                   Fin de message
  .

<--- 250 2.0.0 oBGL0kRC016832 Message accepted for delivery
---> QUIT
<--- 221 2.0.0 paris.ensmp.fr closing connection
Connection to paris.ensmp.fr closed by foreign host.
```

FIG. 2.1: Une transaction entre deux terminaux (client et serveur) de messagerie. Le contenu de la boîte extérieure correspond aux échanges entre les deux terminaux : c'est l'*enveloppe du message*. La boîte de deuxième niveau correspond au *contenu effectif* du message, avec trois composantes : les *en-têtes*, le *corps du message* et une ligne avec un "." pour indiquer la *fin du message*.

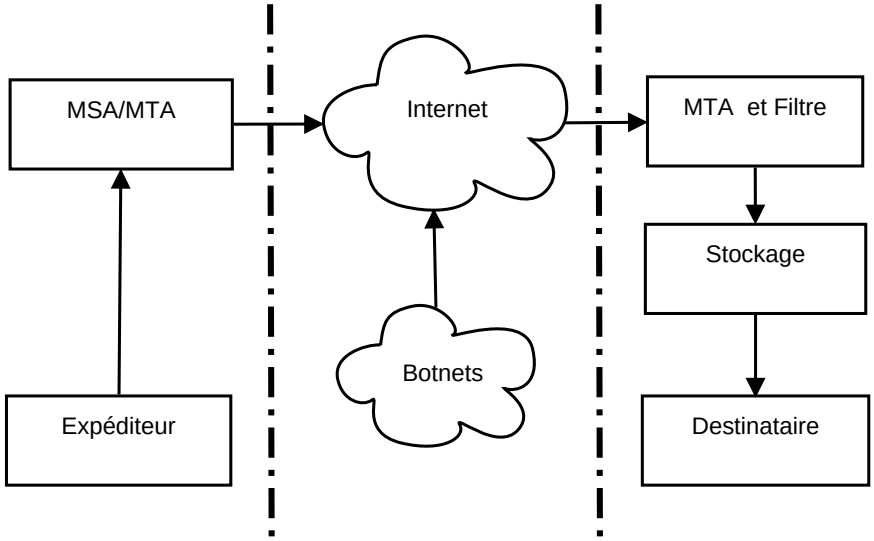


FIG. 2.2: Trajet typique simplifié d'un message - après soumission du message à un MTA (Mail Transport Agent ou "serveur de mail"), celui-ci recherchera son équivalent le plus proche du destinataire (un MX ou Mail eXchanger, qui s'occupera du filtrage en arrivée et enregistrement dans un serveur de stockage de messages.

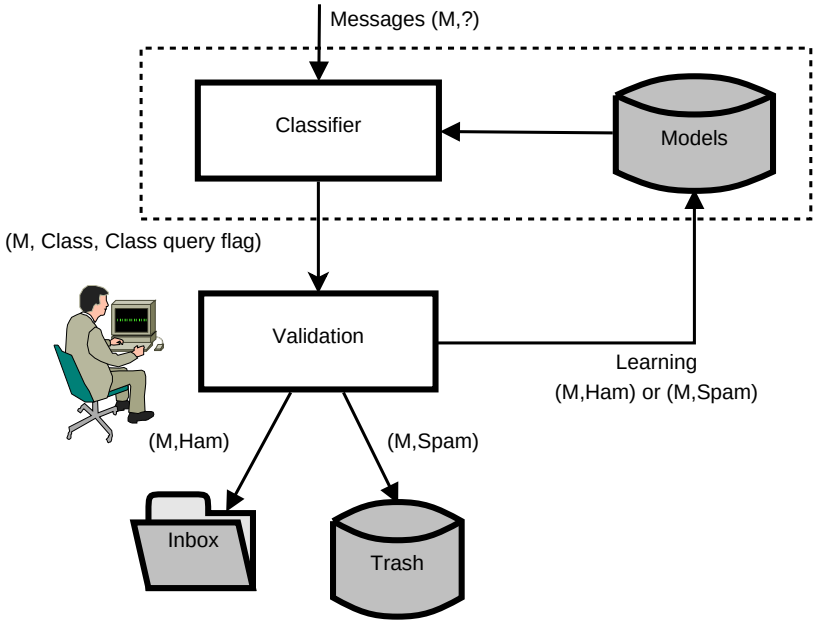


FIG. 2.3: Le processus de classement de messages et les interactions possibles entre le filtre et le destinataire final.

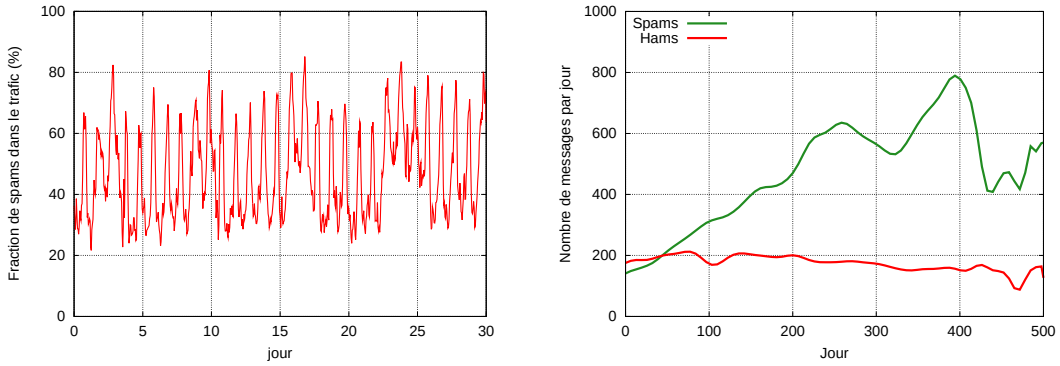
Le destinataire peut aussi retourner des informations au filtre, permettant la mise à jour des modèles utilisés par l'algorithme de classement. Cette particularité caractérise, comme nous le verrons par la suite, le scénario typique d'*apprentissage en ligne*.

2.3.1 Aspects Temporels

Le flot de messages n'est pas stationnaire : ses caractéristiques évoluent en permanence. Il n'est pas aberrant de considérer que les messages légitimes évoluent peu : les expéditeurs changent rarement leurs habitudes d'écriture et, sauf dans certains cas spécifiques, la topologie de leurs réseaux de correspondants reste relativement stable. Le spam, par contre, évolue en permanence, surtout pour déjouer les filtres. Ceci explique, d'une part, le besoin constant de mise à jour des modèles utilisés par les algorithmes de classement et, d'autre part, la nécessité de prendre toujours en compte les messages dans l'ordre chronologique d'arrivée.

Cette dérive impacte les messages de trois façons :

- **La répartition des classes** - La figure (2.4) montre la variation du taux de spam à l'entrée de l'École des Mines de Paris. On peut distinguer l'activité de nuit (pics fins) et de weekend (pics plus larges) qui correspondent aux périodes où il y a une baisse dans les échanges de messages professionnels. Cet exemple montre une variation à court terme, mais on remarque, sur des périodes plus longues (une année, par exemple) une évolution plus ou moins périodique dans les hams liée aux vacances et fêtes et une variation non périodique des spams, plutôt liée à des événements autres (début ou fin d'activité d'un spammeur, débranchement de McColo, événement lié à une célébrité, ...).
- **La répartition des genres à l'intérieur de chaque classe** - ceci est plutôt vrai pour les spams. Rien ne garantit que la répartition par genre (pornographie, arnaques, médicaments, ...) soit constante dans le temps.
- **L'évolution des messages** -Dans la classe *hams*, les expéditeurs changent peu souvent leur façon d'écrire : l'évolution est lente, alors que dans la classe *spams*, pour déjouer les filtres, les spammeurs changent souvent le contenu et la présentation des messages.



(a) Variation journalière du taux de spam à l'entrée du domaine ensmp.fr. Les pics coïncident avec la baisse d'activité professionnelle : nuits et weekends (pics plus larges)

(b) Évolution du nombre de messages, par classe et par jour, reçus par l'auteur sur une période d'un an et demi. La baisse vers le jour 400 correspond à la fermeture de McColo en novembre 2008.

FIG. 2.4: Évolution de la fraction de spams dans le flot de messages à court (1 mois - Fig. 2.4a) et à moyen (1 an et demi - Fig. 2.4b) terme. On remarque la stabilité relative du nombre de messages légitimes par jour, tandis que les spams ont plutôt tendance à augmenter considérablement.

Ces trois aspects montrent que le spam n'est pas un processus statique (stationnaire) : il évolue dans le temps. Les deux derniers illustrent l'importance du respect de l'ordre chronologique des messages et ceci justifie donc que le filtrage de spam soit considéré comme un *processus en ligne* [60].

La compréhension de l'évolution temporelle est particulièrement importante dans l'utilisation partagée d'un filtre anti-spam à cause de la difficulté d'obtenir des échantillons, le but étant de pouvoir les utiliser le plus longtemps possible.

La dérive, à la fois qualitative et quantitative, est due à la génération des messages. Les interactions entre le filtre et le destinataire peuvent aussi, comme nous le verrons par la suite, ajouter des retards dans la boucle d'apprentissage et provoquer une baisse d'efficacité.

2.3.2 Interactions avec le destinataire

Les interactions avec le destinataire sont à double sens : le destinataire retourne des informations vers le filtre et modifie son comportement (c'est l'apprentissage du classificateur) et vice-versa.

Le retour d'information fait par l'utilisateur rend l'apprentissage possible mais c'est une source de complexité dans le modèle : il s'agit d'une action humaine que l'on ne peut pas modéliser avec précision. Citons quelques exemples de comportements humains qui affectent le modèle [59] :

- **Retard** - Le destinataire ne traite pas les messages immédiatement après filtrage, mais à des intervalles qui ne sont pas forcément réguliers allant de quelques minutes à plusieurs heures, voire même jours. Dans un modèle idéal, le classement de chaque message est suivi, immédiatement, de l'information correcte de classement. Ce retour n'étant pas instantané, le modèle sera mis à jour toujours avec un retard aléatoire.
- **Retour partiel** - Le retour d'information peut être systématique ou seulement pour une partie des messages. Les destinataires ont souvent le réflexe de ne renseigner que les messages en erreur, et parfois ils sont plus attentifs au contenu de la boîte légitime et ne signalent, donc, que les spams non détectés.
- **Des Retours d'Information Erronés** - Dans des expérimentations demandant à des utilisateurs humains de classer des messages, des taux d'erreur variant entre 3% et 7% [249] [118] ont été rapportés. D'autres études ont montré que les erreurs ne sont pas uniformément distribuées selon le genre de message, même à l'intérieur de la même classe [151]. Les classificateurs de messages électroniques les plus performants ont des taux d'erreur typiques de l'ordre de 0.5%. Ces erreurs, injectées dans les modèles utilisés par ces classificateurs ne sont pas sans conséquence sur leur efficacité [62].

Le filtre, à son tour, a une influence sur le comportement de l'utilisateur. Plice et al [202] ont suggéré que plus le niveau de spam est faible, plus on est attentif au contenu des spams. En conséquence, une personne regarde plus attentivement une boîte à spams à la recherche de messages légitimes mal classés lorsque le nombre de messages est faible.

2.3.3 Interactions avec l'expéditeur

Il arrive souvent qu'un spammeur souhaite avoir des informations sur le fonctionnement d'un filtre anti-spam, de façon à ce que ses messages puissent le traverser et arriver dans la boîte aux lettres des destinataires, si possible, sans être marqués comme étant du spam.

Lorsque le filtrage se fait par le contenu des messages, les interactions avec l'expéditeur sont indirectes. Un filtre peut soit accepter les spams tout en les marquant comme tel, soit les refuser et, dans ce cas, l'expéditeur peut inférer le résultat du classement. Par contre, si le message n'est pas refusé, l'expéditeur ne peut pas, en principe, estimer le classement du filtre.

Pour pouvoir obtenir des informations plus précises, [179] [180] Lowd et Meek ont imaginé des possibles scénarios d'interaction active dans lesquels l'expéditeur envoie des séquences de messages avec des contenus différents et avec, par exemple, des liens cachés vers des pages web sous son contrôle. La détection d'une consultation de ces pages permet d'inférer que le message est bien arrivé dans la boîte aux lettres du destinataire et a bien été lu. Ainsi, ces méthodes peuvent permettre de connaître assez finement les seuils de détection du filtre. Elles sont utilisées, le plus souvent, par des entreprises de marketing pour évaluer globalement les taux de pénétration de leurs campagnes publicitaires.

2.4 L'apprentissage

L'apprentissage est le processus permettant de construire les modèles, utilisés comme référence par l'algorithme de classement, à partir d'un ensemble d'exemples (ou données d'apprentissage). Un exemple est un couple $(message, étiquette)$. Le but de l'apprentissage est la construction d'une fonction permettant d'associer une *étiquette* à un *message* non vu pendant l'apprentissage.

Ce processus peut prendre des formes différentes selon le type d'algorithme de classement. Dans le cas d'un classificateur *bayésien naïf*, par exemple, il s'agit de compter, pour chaque classe et pour chaque terme du dictionnaire, le nombre de documents où le terme est présent alors que dans le cas d'un classificateur *SVM*, il s'agit de déterminer l'équation d'un hyperplan séparant les deux classes.

On parle d'*apprentissage supervisé* lorsque les modèles sont construits à partir d'exemples dont la classe est connue et d'*apprentissage non supervisé* dans le cas contraire. L'apprentissage non supervisé est utilisé dans les applications de *clustering*, où le but est de regrouper les objets par leur ressemblance, sans connaître, à priori, la classe associée à chaque objet.

On distingue aussi l'*apprentissage en ligne* et l'*apprentissage hors ligne*. Dans l'apprentissage hors ligne (ou *en batch*), les exemples sont entièrement traités dès le départ avant toute opération de classement, tandis que dans l'*apprentissage en ligne* les exemples sont des objets réels à classer et l'apprentissage se fait, au fur et à mesure, grâce au retour d'information concernant les classements qui viennent d'être effectués [239, p.241].

L'apprentissage en ligne a deux caractéristiques qui le rendent particulièrement différent de celui hors ligne : les exemples sont présentés dans un ordre précis (l'ordre chronologique) et le nombre d'exemples utilisés pour l'apprentissage peut ne pas être borné et doit intégrer un dispositif permettant d'*oublier* automatiquement les exemples trop anciens.

Étant donné le caractère évolutif des caractéristiques des messages électroniques, l'apprentissage d'un filtre anti-spam relève typiquement de l'apprentissage en ligne. Malgré cela, nombreux sont les résultats de recherche publiés où les expérimentations et évaluations sont basées sur une hypothèse d'apprentissage hors ligne.

L'*apprentissage* défini le protocole utilisé pour la construction et la mise à jour des modèles utilisés par l'algorithme de classement pour effectuer le filtrage. Cet aspect est critique dans le sens où il doit prendre en compte les interactions entre le filtre et les destinataires et aussi les phénomènes temporels.

2.5 Le filtre

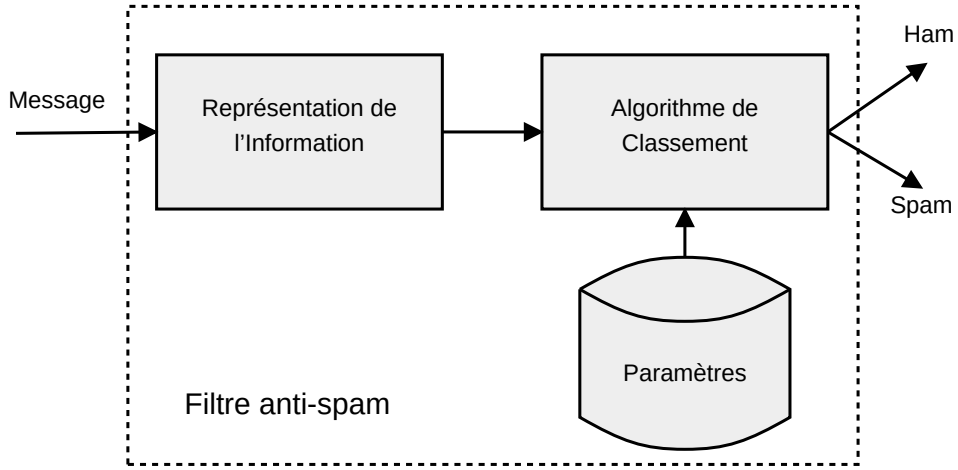


FIG. 2.5: Schéma simplifié d'un filtre anti-spam

Il s'agit de l'élément central du processus. Il est constitué de trois parties (voir Fig. 2.5) :
- **l'algorithme de classement** - c'est la partie "intelligente". Cette partie contient l'implémentation informatique d'une méthode de classement (Bayésien Naïf, SVM, Régression Logistique, Réseau de Neurones, ...) ;
- **la représentation des messages** - Cette partie est chargée d'extraire les caractéristiques (ou attributs) des messages qui seront manipulées par l'algorithme de classement. Les messages bruts sont constitués de suites de caractères qui ne sont pas manipulables directement par les algorithmes de classement ;
- **les paramètres** - c'est l'ensemble des données utilisées par l'algorithme de classement. Ces données sont le résultat du processus d'apprentissage et serviront de référence pour classer les nouveaux messages. Leur forme varie selon le type d'algorithme de classement : les coefficients d'un hyperplan séparateur pour un algorithme du type SVM ou encore les distributions des termes dans chaque classe pour un algorithme du type Bayésien Naïf.

Les *algorithmes de classement* n'ont pas, ou alors très peu, de particularités liées au filtrage de spam. Ce sont des algorithmes utilisés aussi bien pour le traitement d'information textuelle que pour le traitement d'images, de données sismiques ou autres.

L'ensemble *algorithme de classement et paramètres* constitue un *classificateur*.

2.6 La représentation des messages

Les algorithmes de classement ne savent pas, en général, manipuler des objets autres que des objets structurés sous la forme, par exemple, d'un vecteur ou d'une matrice. Les messages électroniques sont des objets textuels non structurés qu'il faut représenter de façon à ce qu'ils puissent être traités par ces algorithmes. La représentation la plus courante est un vecteur où chaque dimension correspond à un *attribut* du message. Il y a deux approches pour définir les attributs des messages.

Dans la première approche, les *attributs* sont définis manuellement : ils correspondent, en général, à la présence de certains mots clés avec un bonne valeur discriminante ou alors à des caractéristiques empiriques, par exemple : *il s'agit d'un message dont le contenu est riche en balises HTML, images, ou liens vers des sites web.* Les attributs définis manuellement sont en nombre fixe et dépassent rarement le millier. Sauf référence à un cas particulier, cette approche n'est pas traitée dans cette thèse : d'une part l'efficacité des classificateurs utilisant ce type de méthode n'est pas concurrentielle [92] et, d'autre part, la messagerie électronique étant un processus non stationnaire, l'évolution exigerait une intervention manuelle pour créer de nouveaux attributs ou pour supprimer ceux devenus inutiles.

Dans la deuxième approche, ce ne sont pas les attributs qui sont définis à l'avance, mais les règles permettant de les extraire. Un exemple de règle serait : *un attribut est une suite de caractères compris entre deux caractères délimiteurs du type espace ou signe de ponctuation.* Cette règle permet l'extraction des mots d'un message. L'ensemble de tous les mots différents trouvés dans l'ensemble d'exemples constitue le *vocabulaire*. Cet aspect dynamique fait que l'ensemble des attributs évolue naturellement sans intervention humaine. L'apprentissage consiste donc non seulement à définir le poids de chaque attribut, mais aussi à construire l'ensemble des attributs.

La communauté de l'apprentissage artificiel utilise le mot *attribut* pour désigner, génériquement, chaque critère ou caractéristique à prendre en compte dans l'objet étudié. Mais lorsqu'il s'agit d'un attribut d'origine textuelle faisant partie d'un vocabulaire, on lui préfère le mot *terme*.

Ainsi, un message est représenté sous la forme d'un vecteur où chaque dimension correspond à un terme du vocabulaire utilisé. La valeur associée à chaque dimension peut indiquer soit la présence/absence du terme dans le message, soit son nombre d'occurrences. Les termes peuvent être des *mots* ou des *n-grams* (suites de n caractères ou mots).

Les messages sont constitués de deux parties : le corps avec le contenu effectif, et les entêtes ou méta-informations avec, par exemple, des informations de format et traçabilité. Le type d'information présent dans ces deux parties n'est pas de même nature et n'a donc pas le même pouvoir discriminant.

Dans la bibliographie concernant le filtrage de spam, il n'est pas rare que cet aspect soit traité de façon superficielle, ou même pas traité du tout, si l'objet principal de la communication est un algorithme de classement. Or, cet aspect est important et on ne peut comparer l'efficacité intrinsèque de deux algorithmes de classement que si les représentations utilisées sont bien précisées et similaires.

Le chapitre 4 est dédié à une discussion plus détaillée sur la représentation des messages dans les applications de classement.

CHAPITRE 3

Historique

Study the past, if you would divine the future.

Confucius

3.1 Introduction

Ce chapitre contient un bref historique des applications de filtrage de spam basées sur l'apprentissage artificiel et domaines connexes.

Ces développements ont été menés de façon indépendante par la communauté de la recherche et celle des développeurs de logiciels libres.

3.2 La communauté de la recherche

3.2.1 Les débuts : Expérimentations avec classificateurs

Les premières publications concernant le classement de messages électroniques basé sur des méthodes d'intelligence artificielle datent de 1996.

Cohen [54] a comparé RIPPER [55] et Rocchio [182, p. 269] pour le classement thématique de messages. RIPPER [55] est un classificateur utilisant un ensemble de règles (présence ou absence des mots du dictionnaire) constitué automatiquement pendant la phase d'apprentissage. Rocchio [182, p. 269] représente le message à classer sous la forme d'un vecteur et évalue la distance (généralement euclidienne définissant la pertinence d'un message) entre ce vecteur et les vecteurs prototypes de chaque classe, associant le message à celle dont le vecteur prototype est le plus proche. Rocchio a son origine dans un les applications d'indexation et recherche documentaire. Les deux méthodes ont présenté des résultats similaires, mais l'auteur a constaté une amélioration lorsque l'apprentissage était fait individuellement pour chaque utilisateur plutôt que globalement pour tous.

En 1998, dans une même conférence, apparaissent les premières publications concernant le filtrage de spam. Pantel [200] et Sahami et al [220] proposent l'utilisation d'un classificateur bayésien naïf [182, p. 234] pour le classement de spams. Sahami n'a retenu, pour l'opération de classement, que les 500 mots les plus significatifs de chaque message. Sahami a constaté que le classement binaire (ham/spam) était plus efficace que le classement multi-classes basé sur le genre du message (pornographie, escroquerie, médicaments, ...). Aussi, des aspects particuliers du filtrage de spam ont été remarqués : la dissymétrie des coûts associés aux erreurs de classement,

ou alors l'utilisation d'attributs synthétiques tels que certaines en-têtes et des mises en forme particulières du message.

L'utilisation de SVMs (Machines à Vecteur de Support) [138] pour le filtrage de spams a été initialement proposée par Drucker et al [92], qui a comparé l'efficacité d'un classificateur SVM linéaire avec RIPPER [55], Rocchio [182, p. 269] et Boosting [104]. Cette publication est intéressante puisque c'est une des premières à avoir essayé un large éventail de configurations d'expérimentation (en particulier sur la construction et la sélection d'attributs ou les modes d'apprentissage). La plupart des conclusions n'ont pas encore été contredites :

- SVM et boosted trees sont comparables, mais les SVMs permettent d'atteindre des taux de faux positifs plus bas plus facilement ;
- Les méthodes basées sur des règles (RIPPER et Rocchio) ne sont pas compétitives pour le filtrage de spam ;
- L'apprentissage des boosted trees est excessivement long ;
- Pour les SVMs, les attributs binaires (présence/absence des termes) donnent de meilleurs résultats alors que les attributs multinomiaux (nombre d'occurrences des termes) sont à privilégier pour les boosted trees ;
- Les procédures de sélection d'attributs constituent des traitements lourds et il vaut mieux les intégrer dans l'apprentissage si on veut les utiliser ;
- Il ne faut pas exclure les termes neutres (stop words).

Plusieurs autres résultats de recherche concernant l'utilisation de SVMs pour le filtrage de spam ont été publiés par la suite. Kolcz [148] a étudié la prise en compte des erreurs de classification spécifiques à chaque classe. Islam [131] a proposé une méthode de sélection d'attributs. Malgré l'efficacité constatée, les SVMs restent des algorithmes non triviaux à mettre en oeuvre et consommateurs de ressources. Des implémentations efficaces ont été proposées, par exemple, par Joachims [137] [139] et Bordes [29], pour des SVMs linéaires dans un contexte général de classement. Pour l'apprentissage et le classement en ligne de spams, Sculley [229] a proposé l'utilisation de ROSVMs (Relaxed Online SVMs), une simplification limitant le nombre d'itérations de l'algorithme d'optimisation et d'exemples avec une efficacité de classement qui restait encore proche de celle que l'on peut obtenir sans simplification.

Androutsopoulos et al ont évalué et comparé le classificateur bayésien naïf [8], le classificateur à mots-clés [7] et k-NN (les k voisins les plus proches) [9] [221] explorant la sensibilité des méthodes à différentes variantes de configuration telles que la lemmatisation[1] des termes, le nombre d'attributs ou le nombre d'exemples. Les métriques d'évaluation d'efficacité ont intégré des coûts différents aux erreurs de classement selon la classe. Globalement, sauf pour le classificateur à mots-clés, les résultats ont été équivalents. Une conclusion intéressante, déjà à l'époque, a été de constater que le filtrage de spam est un problème plus ardu que le problème usuel de classement de messages électroniques. Les résultats obtenus par Sakkis et Androutsopoulos [221] avec k-NN étaient meilleurs avec des listes de diffusion qu'avec une messagerie individuelle, ce qui pourrait s'expliquer par l'homogénéité des messages dans ce type de corpus.

Cependant, concernant le corpus de messages utilisé par Androutsopoulos pour l'apprentissage et les tests, quelques remarques sont nécessaires. Ce corpus, *LingSpam*[2], est constitué seulement de 481 spams et 2412 hams. Sa taille n'est pas suffisante pour valider un dispositif dont les taux d'erreur peuvent être inférieurs à 1 %. Les messages légitimes viennent tous d'une liste de diffusion consacrée à la linguistique : un ensemble trop homogène. De même, les fichiers attachés et les balises de formatage (HTML) ont été supprimés. Ces remarques suggèrent que le corpus n'est pas représentatif d'un flot réel de messagerie et que ces résultats doivent être analysés avec précaution. Des remarques similaires sont valables pour les corpus PU1, PU2, PU3 et PUA[3] utilisés, comme LingSpam, pour les expérimentations de plusieurs travaux de recherche. Malgré cela, ces corpus ont eu le mérite d'exister et ont permis d'avoir un repère de comparaison,

[1] *Lemmatisation* : opération regroupant en une seule entité canonique (le *lemme*) les mots d'une même famille ou les différentes formes (le nom, le pluriel, le verbe à l'infinitif, ...)

[2] LingSpam : `http://labs-repos.iit.demokritos.gr/skel/i-config/downloads/`

[3] *PU1, PU2, PU3 et PUA* : `http://labs-repos.iit.demokritos.gr/skel/i-config/downloads/`

valide à l'époque.

Sasaki et Shinnou [223] ont proposé le clustering (apprentissage non supervisé) pour regrouper les exemples selon leur similarité, attribuant à chaque cluster la classe *spam* si plus de 85 % des messages étaient du spam et la classe *ham* sinon. Le classement des nouveaux messages s'effectue par comparaison avec les centroïdes des clusters. L'efficacité annoncée par l'auteur (équivalente à celle des SVMs et meilleure que celle de Bogofilter[4]) est à prendre avec précaution à cause de l'utilisation du corpus *LingSpam*. Ces conclusions sont même infirmées par des résultats obtenus par ailleurs, par exemple, dans TREC. Aussi, le seuil d'association d'un cluster à une classe ne semble pas compatible avec le taux d'erreur inférieur à 1 %, attendu dans une application de classement de spam. Néanmoins, cette proposition semble intéressante pour effectuer un classement préliminaire lors de la constitution d'un corpus destiné à l'apprentissage ou l'évaluation d'un filtre.

Carreras et Marques [43] ont utilisé Boosting Trees (AdaBoost) pour le filtrage de spams et comparé leurs résultats avec ceux du classificateur bayésien naïf de Androutsopoulos [8] sur le même corpus, et ces derniers se sont avérés meilleurs. Carreras a remarqué que les erreurs de classement concernaient, le plus souvent, des cas où la confiance dans le résultat était faible (score proche de la valeur neutre de classement).

Zhang et Yao [163] ont proposé l'utilisation d'un modèle d'entropie maximale (régression logistique) pour filtrer le spam. Dans leur approche, ils ont utilisé un modèle mixte ajoutant aux attributs textuels usuels (mots) les 10 règles fixes les plus significatives trouvées dans leur corpus par SpamAssassin[5] (logiciel libre, décrit dans la Section 3.3.1). Les résultats obtenus ont été legèrement meilleurs que ceux obtenus avec un classificateur bayésien naïf. L'apprentissage du modèle a été fait avec 100 itérations de l'algorithme GIS (Generalized Iterative Scaling) [84], dont la lourdeur est le principal inconvénient pour un filtrage en ligne.

Goodman et al [115] ont proposé une autre version de classificateur à régression logistique avec utilisation d'un algorithme de descente de gradient comme alternative à GIS pour l'apprentissage. Cette version de classificateur est non seulement plus rapide et donc plus adaptée à l'apprentissage en ligne, mais aussi une des plus efficaces au moment de la rédaction de cette thèse.

L'utilisation de modèles statistiques de compression de données pour le filtrage de spams a été proposée par Bratko [35] [36] et Cormack [59, p. 67] avec de très bons résultats lors de la conférence TREC 2006.

SpamGuru [232] [233], développé par le département de la recherche d'IBM, combine les résultats de plusieurs algorithmes de classement (qui ne sont pas tous statistiques) placés en série. Il s'agit d'une solution globale d'entreprise, permettant d'utiliser des préférences globales et individuelles, en choisissant la plus spécifique disponible.

Lynam et Cormack [181] ont combiné de plusieurs façons (méta-classificateurs) les filtres présentant les meilleurs résultats lors des expérimentations de TREC. Les résultats obtenus étaient meilleurs que le meilleur des filtres participants.

Sculley [227] a proposé l'utilisation de "mini-filtres", des filtres n'utilisant que quelques dizaines d'attributs. Les résultats obtenus sont bons mais doivent encore gagner en robustesse. Selon Sculley, ce type de filtres trouverait une application en tant que complément de personnalisation au résultat d'un filtre global.

3.2.2 Le filtrage de spam : un problème à part entière

En 2003, Tom Fawcett a publié un article [100] dans le journal KDD Explorations du groupe SIGKDD de l'ACM, dans lequel il faisait le point sur la problématique du spam. Jusque là, la recherche sur le spam était abordée comme une application particulière de domaines tels que Machine Learning, Information Retrieval, Data Mining et Statistique. Mais Fawcett a énuméré une série de points qui font du spam un problème à part. Sa conclusion était que le problème

[4] *Bogofilter* : `http://bogofilter.sourceforge.net`
[5] *SpamAssassin* : `http://spamassassin.apache.org`

du spam ne pourrait pas être résolu par les seules techniques habituelles de ces domaines mais, qu'en même temps, son étude pourrait bénéficier à tous.

Contrairement aux applications courantes, le spam est une "course aux armements" [100] [179] [13] : celui qui envoie les spams essayant sans cesse de surpasser celui qui les détecte. Ce peut être, par exemple, en insérant des mots couramment trouvés dans les messages légitimes (Good Words) pour éviter que le message soit classé comme spam [82] [179] [180] [275]. Une autre astuce courante consiste à remplacer les caractères des mots par d'autres caractères ayant des formes similaires, de façon à rendre difficile l'extraction des attributs [127] [167] : par exemple, **V|4@r4** au lieu de *VIAGRA*.

Pu et Webb [204] ont étudié l'évolution des stratégies de création de spams sur une période de 3 ans. Pour cela, ils ont utilisé le logiciel libre SpamAssassin et noté à quel moment chaque règle devenait active ou obsolète pour la détection des spams. Cela leur a permis de démontrer empiriquement que la génération de spams n'est pas un processus statique et que, pour ce type d'algorithme de classement, certaines règles gardent leur efficacité plus longtemps que d'autres.

Le spam n'étant pas un processus stationnaire, les caractéristiques statistiques ne sont pas constantes dans le temps. Ainsi, tout traitement d'un ensemble de messages effectués pour l'apprentissage ou l'évaluation d'efficacité doit prendre en compte l'ordre chronologique des messages. Donc, les méthodes habituelles utilisées dans un contexte de classement hors ligne, telles que *K-fold cross-validation* [5, p. 300], ne sont pas applicables à l'évaluation d'efficacité des classificateurs [59, p. 75].

Les modèles utilisés par les classificateurs doivent être mis à jour de façon continue. Delany, Riverola, Hsiao et autres [87] [86] [102] [129] ont étudié l'utilisation d'une fenêtre temporelle glissante pour sélectionner les messages à utiliser pour l'apprentissage.

Cormack et Martins [71] [81] ont étudié l'effet de l'âge absolu et relatif des exemples utilisés lors de l'apprentissage sur l'efficacité d'un classificateur à régression logistique et ont comparé l'apprentissage incrémental avec l'utilisation d'une fenêtre temporelle glissante. Ils ont montré empiriquement qu'avec la suppression des références temporelles, la perte d'efficacité dans le temps d'un filtre dont l'apprentissage n'est pas mis à jour est supportable dans certaines limites. Ces résultats seront présentés dans le chapitre 8.

L'asymétrie du coût des erreurs de classement est une autre particularité du filtrage de spam. Androutsopoulos et al [10] ont proposé l'utilisation d'un paramètre de pondération pour corriger l'erreur globale de classement. Cette erreur serait évaluée selon :

$$erreur\ corrigée = \frac{\lambda \cdot FS + FH}{\lambda \cdot FS + \lambda \cdot VH + VS + FH} \tag{3.1}$$

(F et V pour *faux* et *vrai* et H et S pour *ham* et *spam*). Hidalgo [128] a considéré que s'il est vrai que le coût des erreurs est asymétrique, rien ne permet d'évaluer l'importance du coût du mauvais classement d'un message légitime, par rapport à l'erreur inverse. Cormack [69], complétant les arguments de Hidalgo, estime que, finalement, il vaut mieux laisser ce jugement à la discrétion du destinataire des messages. Avec une autre approche, Kolcz et Alspector [148] ont proposé la prise en compte de l'asymétrie dans l'apprentissage d'un classificateur SVM, en attribuant des poids différents selon le genre à l'intérieur de chaque classe.

3.2.3 La contribution de TREC - Spam Track

Pendant trois ans, la *Text REtrieval Conference (TREC)* du NIST[6] a hébergé le groupe *Spam Track*. Le but de cette conférence est de promouvoir la recherche de nouvelles méthodes de traitement de documents textuels à grande échelle, avec la particularité de faire travailler ensemble, sur un même thème, tous les participants de chaque groupe de travail (*Track*). Ce fut ainsi le cas pour le spam.

La contribution la plus importante de TREC n'est pas tant dans ses résultats que dans l'établissement d'un cadre commun d'évaluation des classificateurs. Avant TREC, les classificateurs

[6]TREC - http ://trec.nist.gov

étaient développés, évalués et comparés avec des critères et des ensembles d'exemples choisis par les auteurs des travaux eux-mêmes. De ce fait, les environnements d'évaluation n'étaient pas toujours réalistes et la comparaison des différents résultats publiés était souvent difficile.

Constitution d'un corpus de messages

La constitution d'un corpus de messages a été l'un des travaux préliminaires de TREC. Jusque là, les classificateurs étaient évalués avec des messages de provenances diverses, en général peu représentatifs d'une vraie boîte aux lettres et en quantité insuffisante pour que les résultats soient significatifs.

Les corpus de TREC [65] [68] [70] ont été constitués à partir des messages légitimes reçus par les employés de l'entreprise Enron dans les 3 mois autour de sa faillite [64]. Un certain nombre de spams ont été ajoutés aux spams d'origine pour compenser ceux qui avaient été supprimés par les destinataires. Ce corpus contient 92189 messages, dont 39399 hams et 52790 spams. Outre les mesures d'efficacité faites avec un meilleur degré de confiance, la taille du corpus a aussi rendu possible d'autres évaluations telles que la vitesse d'apprentissage.

Le corpus utilisé en 2007 a été le premier corpus, toujours synthétique, dont les messages des deux classes ont été reçus sur le même serveur de messagerie et à la vraie date.

Métriques d'évaluation de filtres

Cette contribution résulte des travaux préliminaires de Gordon Cormack [69] sur l'évaluation de filtres anti-spam. Dans ses travaux, Cormack a utilisé un corpus de hams et de spams collectés pendant 8 mois et a comparé l'efficacité de 6 filtres distribués sous licence libre.

L'étude de toutes les métriques usuelles utilisées dans les problèmes de classification (Machine Learning, Information Retrieval, diagnostique médical, ...) a permis de sélectionner celles qui seraient les plus pertinentes pour le problème du classement de spam : HMR (ham misclassification rate ou taux de faux positifs), SMR (spam misclassification rate ou taux de faux négatifs), LAM (Logistic Average Misclassification), ROC (Receiver Operating Characteristic) et ROCA (ROC Area). Ces résultats confirment des travaux antérieurs de Cormack [69] [67] qui démontraient que, pour des applications du type classement d'objets textuels, ces métriques étaient plus pertinentes que celles habituellement utilisées en recherche documentaire. Typiquement, il n'est pas rare que des travaux de recherche soient évalués par *précision* et *rappel* (*precision* et *recall*), métriques pertinentes en recherche documentaire [182], mais pas en classement de messages électroniques.

Modèle de classement et évaluation en ligne

Les méthodes utilisées précédemment pour évaluer les filtres étaient celles utilisées dans des domaines où les caractéristiques du flot d'information ne changent pas dans le temps (processus stationnaire). Ce n'est pas le cas du spam. Il est important de présenter les messages dans l'ordre chronologique, et les méthodes d'évaluation adaptées aux applications hors ligne, *e.g. K-fold cross-validation* [5, p. 300], présentent donc peu d'intérêt.

TREC a permis aussi d'évaluer différentes configurations d'apprentissage en ligne : retour immédiat des erreurs, retour d'erreur avec retard ou encore apprentissage actif.

Une boîte à outils a été développée pour mettre en pratique cette méthodologie. Elle permet d'envoyer les commandes au filtre : *initialize*, *learn*, *classify*, *end*. Le résultat, constitué de l'ensemble des métriques d'évaluation, est présenté à la fin de l'opération de test. Cette boîte à outils est livrée (pour TREC) avec les interfaces pour Bogofilter, CRM114, DSPAM, dbacl, Popfile, Spamassassin, SpamBayes et Spamprobe.

Résultats

Chaque année, les filtres les plus efficaces ont présenté des résultats bien meilleurs que ceux de l'année précédente. En 2005, Bratko a montré que les modèles à compression étaient bien meilleurs

que les filtres pseudo-bayésiens naïfs de l'époque [66]. En 2006, Assis et Siefkes, en utilisant des bigrammes orthogonaux épars et un apprentissage sur seuil, ont obtenu les meilleurs résultats [56], mais il a été démontré par la suite que cette méthode était sur-ajustée et pratiquement inutilisable en présence de bruit [226]. En 2007, Sculley a prouvé que les SVMs pouvaient être adaptés au filtrage avec apprentissage en ligne, alors que Cormack a obtenu une efficacité comparable avec un filtre à régression logistique avec apprentissage par descente de gradient [57]. Cormack et Sculley ont amélioré les résultats de l'approche par compression, en utilisant, comme attributs, des n-grams de niveau caractère, au lieu des mots habituellement utilisés.

Selon Gordon Cormack[7] :

> *All of the best approaches have a remarkable characteristic : they are completely general purpose methods. Spam- and email-specific engineering don't seem to contribute to the performance of the best filters – at least nobody has figured out how.*

3.3 Les praticiens et développeurs de logiciels libres

3.3.1 Les classificateurs à règles fixes

Le premier logiciel de classement de messages est apparu en 1990 : *procmail*[8]. Le but n'était pas le filtrage de spam, mais le classement thématique. Son fonctionnement est défini par un ensemble de règles fixes et simples (expressions régulières) appliquées, de façon séquentielle, sur le corps ou les en-têtes des messages. A chaque règle correspond une action particulière, déclenchée lorsque la règle correspondante est satisfaite. Par exemple, la règle suivante dit que si l'en-tête *X-j-chkmail-Status* contient le mot *HI*, le message est ajouté au dossier *SPAM-HI* :

```
:0
* ^X-j-chkmail-Status:.*HI
SPAM-HI
```

Bien que d'une grande simplicité, ce logiciel est encore largement utilisé, les règles de classement ne constituant qu'une partie de ses fonctionnalités.

Une dizaine d'années plus tard, est apparu *SpamAssassin*[9], un filtre utilisant un ensemble de règles fixes (environ 700, dans sa version actuelle[10]) définies manuellement par les auteurs du logiciel. Ces règles vérifient par exemple si la mise en page du message est riche, si le texte est de plusieurs couleurs, si des images sont incluses dans le texte, ou encore si le message mentionne des montants en millions de dollars... On peut aussi définir des règles externes telles que le score attribué par un classificateur statistique ou la présence de liens vers des sites web figurant dans des listes noires (telles que SURBL[11]). Le poids associé à chaque règle est défini au départ à l'aide d'un réseau de neurones. En mode classement, le filtre attribue un score qui est la somme des poids des règles satisfaites par le message en examen et le déclare comme étant du spam ou du ham si le score dépasse ou non un certain seuil. SpamAssassin est un filtre distribué avec une licence GPL et est très largement utilisé.

3.3.2 Les classificateurs adaptatifs (avec apprentissage)

Malgré les résultats prometteurs des travaux de recherche publiés depuis 1998, très peu d'applications ont vues le jour et sont souvent restées confinées dans l'entourage des auteurs, généralement dans des organismes de recherche. Nous citerons, par exemple, *ifile* [206], filtre anti-spam développé au MIT et basé sur un classificateur bayésien naïf.

[7] Gordon Cormack : correspondance personnelle
[8] *Procmail* : `http://www.procmail.org`
[9] *SpamAssassin* : `http://spamassassin.apache.org`
[10] Liste de règles de SpamAssassin : `http://spamassassin.apache.org/tests.html`
[11] *SURBL* : `http://www.surbl.org`

En 2002, dans un essai publié sur son blog, Paul Graham [116] [117] proposait l'utilisation d'un classificateur bayésien naïf pour identifier les spams et incluait un exemple de réalisation du filtre en langage LISP. Contrairement aux publications scientifiques précédentes, cet essai est arrivé au moment opportun et n'est pas passé inaperçu dans la communauté du logiciel libre. Yerazunis [267] a sorti un filtre statistique similaire presque au même moment.

De nombreuses implémentations sont apparues rapidement : *Bogofilter*, *DSPAM*, *CRM114*, *SpamOracle*, *SpamBayes*, *SpamAssassin*, ou encore les filtres intégrés dans la suite *Mozilla/Thunderbird*. Ces produits, distribués avec une licence GPL, sont largement utilisés et ont permis de valider, en vraie grandeur, la méthode de filtrage.

Des expérimentations ont pu être réalisées sur des installations réelles avec du trafic réel. Même si la plupart des résultats n'ont été publiés que sur des pages web, ou s'ils ont été parfois obtenus avec une rigueur insuffisante, ils apportent des contributions et des pistes de recherche intéressantes.

Une des voies d'expérimentation a porté sur la façon de segmenter un message. La majorité des filtres utilisent, comme attributs du classificateur, des mots ou des couples de mots. *CRM114* [242] [268] utilise des combinaisons de deux mots dans une fenêtre glissante de cinq mots, avec attribution de poids plus forts aux combinaisons avec un nombre de mots plus important. Les auteurs désignent cela par les expressions "Orthogonal Sparse Binary Hash" (OSPB) et "Discrimination Markovienne", alors que la prise en compte de la dépendance entre un mot et les précédents est la seule similitude entre ce procédé et les modèles Markoviens usuels.

On a aussi étudié les différents ensembles de caractères délimiteurs pour identifier le plus adapté à la segmentation des messages. L'intérêt réside dans le caractère antagonique du filtrage de spam : une des façons de tromper les classificateurs est de rendre hasardeuse la segmentation des messages et donc l'extraction des attributs.

D'autres suggestions ont été faites sur la façon d'estimer la vraisemblance du message, connaissant la classe, à partir des probabilités individuelles des attributs (au lieu du produit habituel). Dans un premier essai, Robinson [214] propose d'évaluer non pas la vraisemblance, comme proposé par Paul Graham, mais une sorte de combinaison des moyennes géométriques ou bien d'utiliser un test de χ^2. Greg Louis [178] a montré empiriquement que ces suggestions permettaient d'améliorer l'efficacité du filtre *Bogofilter*. D'autres heuristiques, toujours empiriques, ont été proposées et décrites par Zdziarski [273].

Ces différentes expérimentations ont produit des filtres plus efficaces mais, à vrai dire, aucun (ou très peu) d'entre eux ne mérite la désignation de *bayésien naïf* qui leur est attribuée. La raison du succès de ces filtres est la simplicité de mise en oeuvre, la robustesse et l'efficacité atteignable avec peu d'efforts.

3.4 Discussion, Controverses et Conclusions

Pendant longtemps les communautés de la recherche académique, des logiciels libres et des produits commerciaux ont suivi chacune leur chemin indépendamment des autres. La communauté des produits propriétaires a soit utilisé les résultats issus de la communauté des logiciels libres, soit mené leur propre activité de recherche.

Un certain nombre de situations controversées et parfois même conflictuelles ont pu exister entre personnes de communautés différentes et on peut retrouver leur trace à l'aide des moteurs de recherche usuels. On peut les comprendre puisque ce sont des communautés avec des démarches scientifiques assez différentes. D'autre part, la messagerie et sa problématique du spam a été traitée, avant tout, par une communauté de praticiens en informatique avant d'intéresser la communauté scientifique. De la lecture des résultats de recherche les plus anciens, on peut remarquer que la communauté scientifique s'intéressait au spam plutôt comme terrain d'expérimentation de leurs recherches en cours, tandis que la communauté de praticiens le vivait à plein temps.

TREC Spam Track a été l'occasion pour faire le point. TREC a pu, d'une part, clarifier le contexte de fonctionnement d'un filtre anti-spam, proposer et valider une méthodologie d'éva-

luation de ces filtres et, d'autre part, comparer l'efficacité des filtres issus des communautés des logiciels libres et de la recherche : des communautés plus ouvertes à la comparaison d'efficacité. La première année, les filtres dits "bayésiens naïfs" ont été les meilleurs mais, par la suite, ils ont été détrônés par les filtres à compression et finalement par les SVM et régression logistique. Aucun système de filtrage commercial, autre que celui d'*IBM* présenté par leur département de recherche, n'a participé à *TREC*.

Ces expérimentations ont aussi démontré que l'on peut atteindre, avec peu d'efforts, des indices d'efficacité très bons, et que ces indices peuvent encore être améliorés grâce à des algorithmes plus complexes tels que SVM et régression logistique. Néanmoins, la suppression des erreurs résiduelles est pratiquement impossible (ceci sera discuté dans le chapitre 12).

Le nombre de publications dédiées au filtrage de spam parues ces 10 dernières années est de l'ordre de quelques milliers. Néanmoins, moins de la moitié apportent un éclairage intéressant. Parmi les publications de synthèse, on peut citer Guzella [123] et surtout Cormack [59].

Enfin, avec un peu de scepticisme sur la capacité de résoudre la problématique du spam avec des méthodes de fouille de données, Tom Fawcett [100] énumère les challenges du problème et encourage cette communauté à le traiter estimant que cela ne peut que être profitable à la recherche dans ce domaine.

Fawcett [100] décrit le problème de filtrage de spam, et les challenges :

- *La distribution à priori des classes est évolutive et asymétrique ;*
- *Le coût des erreurs est asymétrique et incertain ;*
- *Motifs textuels complexes nécessitant des analyseurs textuels sophistiqués ;*
- *La définition des classes est floue, avec des phénomènes partagés et caractéristiques temporelles complexes ;*
- *Adversaires intelligents et adaptatifs.*

Deuxième partie

Les briques d'un classificateur de messages électroniques

CHAPITRE 4

La Représentation des Messages Électroniques

Everything should be as simple as it is, but not simpler.

Albert Einstein

4.1 Introduction

Dans les problèmes de classement d'objets textuels, la représentation des objets est un aspect aussi important que le choix de l'algorithme de classement.

Un message électronique est un objet semi-structuré avec deux parties définies par des normes techniques tels la RFC-2822 [207] :

1. Des méta-informations structurées, les en-têtes (headers), avec des champs tels les adresses de l'expéditeur, du destinataire, la date, un identificateur unique, le type de contenu, ...
2. Le corps du message, au format ASCII, avec le contenu effectif du message et les éventuels fichiers attachés.

Cette structure est suffisante pour les messages dont le seul but est de transmettre des messages textuels codés en ASCII (7 bits - caractères non accentués). Lorsqu'on souhaite envoyer des fichiers attachés, des messages utilisant une codification des caractères autre que le simple ASCII, un contenu multimédia (image, vidéo, ...) ou encore d'autres contenus plus complexes, on peut utiliser la structure définie dans la RFC-2045 [103]. Dans cette extension, le corps du message est, à son tour, à nouveau organisé selon l'objet de niveau plus haut (le message), avec une partie en-tête et un corps. Des nouveaux champs pour la partie structurée sont ajoutés permettant d'identifier le contenu, l'organisation et l'encodage de la partie non structurée. Il s'agit, en fait, d'une organisation hiérarchique permettant une manipulation aisée par des algorithmes récursifs.

La plupart des algorithmes de classement attendent trouver l'objet à classer sous une forme structurée comme, par exemple, un vecteur dont les composantes sont les *attributs*. Ce n'est pas la représentation habituelle des documents textuels et encore moins des messages électroniques.

Les options retenues pour représenter et structurer les messages et le choix des attributs impactent très fortement sur l'efficacité du classement. On doit être sceptique par rapport à toute publication qui se concentre sur les algorithmes de classement et qui donne peu d'importance à la représentation utilisée ou au mode d'apprentissage.

Autres les messages, nous avons aussi besoin de représenter autres objets dérivés des messages :

* **Boite aux lettres** - c'est l'ensemble (ou un sous ensemble) de messages appartenant à un destinataire unique, ou à un groupe de destinataires.
* **Corpus** - c'est un ensemble de messages utilisé comme référence d'étude, soit pour l'apprentissage, soit pour le classement.
* **Dictionnaire ou Vocabulaire** - c'est l'ensemble des termes (ou attributs) que l'on peut trouver dans une boîte aux lettres (ou un corpus).

Dans ce chapitre nous présentons les méthodes généralement utilisées pour extraire les *attributs* et structurer des documents textuels avec les spécificités des messages électroniques.

4.2 Génération et Représentation des Messages

4.2.1 Génération des Messages

Dans les applications de traitement statistique de textes, le modèle du processus de génération des messages est un aspect important, en particulier pour les algorithmes génératifs tels le bayésien naïf. Le modèle *Bag Of Words* (ou *Sac De Mots*) est le modèle le plus souvent utilisé pour représenter ce processus. Dans ce modèle, un message est le résultat d'une suite de tirages aléatoires indépendants avec ou sans remise, des mots (ou plutôt *termes*) d'un vocabulaire, d'où l'expression *bag of words* ou *BOW*.

$$m = (m_1, m_2, \ldots, m_L),\ m_i \in \mathcal{V} \tag{4.1}$$

où L est la longueur du message.

Ce modèle peut supposer deux hypothèses :

1. Toutes les permutations possibles des termes d'un message ont la même représentation : l'ordre des termes n'a pas d'importance, seule leur présence compte ;
2. L'occurrence d'un terme quelconque est statistiquement indépendant de l'occurrence de n'importe lequel autre terme, sachant la classe d'appartenance du message.

La première hypothèse est admise par quasiment tous les algorithmes de classement. La deuxième hypothèse est celle que l'on trouve dans la bibliographie comme étant *"l'hypothèse d'indépendance statistique des termes"* ou *"hypothèse bayésienne naïve"* [90] [126] [152] et c'est d'ailleurs la raison de la qualification *"Bayes Naïf"* de cet algorithme, qui l'admet explicitement dans l'estimation de la vraisemblance, comme nous verrons dans le chapitre 6.

4.2.2 Représentation

Un message est, dans la forme la plus couramment utilisée par les algorithmes de classement, représenté par un couple $(\mathbf{x}, y)$. $\mathbf{x}$ est un vecteur[1] de N attributs :

$$\mathbf{x} = (x_1, x_2, ..., x_N) \tag{4.2}$$

Les indices des x_i peuvent correspondre à l'ordre du *terme* dans le vocabulaire et la valeur associée (un nombre, une catégorie ou un booléen) sert à quantifier l'évidence de x_i sur le message m : présence ou absence du terme ou alors le nombre d'occurrences. Dans ce type de représentation, N est la dimension du vocabulaire ($N = |\mathcal{V}|$).

$y \in \{ham, spam\}$[2] est l'étiquette associée au message, indiquant la classe (ou catégorie) d'appartenance. La valeur de y est connue pour les messages de l'ensemble d'apprentissage et à déterminer pour les messages à classer.

[1] Cette représentation est aussi connue par le nom *Vector Space Model* [222] ou *VSM*

[2] Des représentations courantes pour les catégories de l'étiquette sont des valeurs booléens $\{faux, vrai\}$, ou encore $\{0, 1\}$ ou $\{-1, 1\}$

Ce sont, bien entendu, des vecteurs creux, puisque le nombre de termes différents dans un vocabulaire est, généralement, bien plus important que la diversité des termes trouvés dans un message. Cette représentation sous la forme vectorielle est plutôt théorique puisque, d'une part, ce sont des vecteurs creux très faiblement peuplés et, d'autre part, la taille d'un vocabulaire est généralement importante et inconnue. En pratique, les messages sont représentés sous la forme d'une liste de couples (attribut, valeur) et le vocabulaire est enregistré dans une mémoire associative telle une base de données.

La *segmentation* est l'opération d'extraction des attributs du message et construction du vecteur le représentant. A chaque attribut différent on associe une valeur : un booléen pour indiquer la présence de l'attribut dans le message (tirage sans remise), un entier indiquant le nombre d'occurrences (tirage avec remise) ou encore une valeur indicative de pertinence.

L'expression *"sac de mots"* représente le mode de génération des messages plutôt que le type d'attribut, qui peut être de n'importe lequel et pas forcément un *mot* dans le sens linguistique. Par la suite nous utiliserons le mot *attribut* pour nous référer aux composantes du vecteur ou parfois *terme*, lorsqu'il s'agit d'un attribut extrait par un modèle linguistique.

La perte d'information due à l'hypothèse d'indépendance statistique des termes fait que la représentation d'un message est irréversible : le message de départ ne peut plus être ni restauré à partir de la représentation vectorielle, ni identifié - plusieurs messages peuvent être représentés de la même façon puisque toutes les permutations des termes sont équivalentes. Cela peut sembler décevant mais la représentation doit contenir l'information juste suffisante pour que le classificateur puisse identifier la classe du message : préserver l'information de départ n'est pas un besoin.

4.2.3 Vocabulaire et Dictionnaire

On utilise souvent indistinctement ces deux mots. Il s'agit de l'ensemble de termes utilisés dans l'application de classement. Il est important de remarquer qu'il ne s'agit pas des mots dans le sens usuel de la vie courante : un terme peut-être un mot, une suite de mots, une phrase, une date ou encore une suite d'un nombre fixe (n) caractères ou mots (un *n-gram*).

Une représentation courante d'un vocabulaire est sous la forme d'un vecteur, avec une composante par terme et dont la valeur associée à chaque composante dépend du type d'algorithme de classement utilisé. Dans une des représentations courantes, on associe à chaque composante sa probabilité dans chacune des classes.

Dans les applications courantes de classement de *spam* (comme nous verrons dans les chapitres suivants), la dimension d'un vocabulaire varie, selon le type de *forme graphique* utilisée dans la segmentation, entre quelques centaines de milliers à quelques millions.

4.2.4 Boîtes aux lettres

Des boîtes aux lettres sont des ensembles de messages : pour un destinataire, pour un thème d'un destinataire, ou un groupe de destinataires. Les représentations usuelles sont :

– Une séquence d'objets (messages) dans leur format d'origine :

$$M = \{m_1, m_2, \dots\}$$

– Une séquence d'objets (messages) *segmentés* où chaque élément $\mathbf{x_i}$ est la représentation vectorielle (attributs extraits) du m_i correspondant ;

$$X = \{\mathbf{x_1}, \mathbf{x_2}, \dots\}$$

– Une représentation probabiliste où chaque élément représente, par exemple, la fréquence de chaque terme dans l'ensemble des messages :

$$C = \{c_1, c_2, \ldots, c_{|\mathcal{V}_M|}\}$$

et $\mathcal{V}_M$ est le vocabulaire des termes trouvés dans l'ensemble des messages de la boîte aux lettres.

4.3 Segmentation du texte : approche linguistique

La *segmentation*[3] est l'opération qui consiste à décomposer un texte en *unités minimales* : des unités qui ne seront pas décomposées d'avantage. L'unité naturelle de décomposition est souvent le *mot* (ou plutôt la *forme graphique*).

Joachims [138, p. 12] classe les méthodes de segmentation, selon le niveau de complexité, en :

* **Niveau *sub-mot*** - décomposition de mots selon leur morphologie.
* **Niveau *mot*** - les mots avec, éventuellement, information lexicale.
* **Niveau *mots multiples*** - groupe de mots, phrases avec, éventuellement, information syntaxique.
* **Niveau *sémantique*** - compréhension du texte.
* **Niveau *pragmatique*** - compréhension du texte avec prise en compte de son contexte.

Quelque soit le niveau choisi, le message est représenté par un vecteur dont les termes (attributs) non nuls sont ceux présents dans le message.

4.3.1 Niveau mots (ou formes simples)

Il y a un consensus sur l'utilisation des *mots* comme unité de base dans les applications de recherche documentaire et classement de textes [138] [213] [231] : il s'agit d'une décomposition naturelle, d'implémentation simple et dont l'efficacité a été démontrée. Même quand le niveau de décomposition est plus élevé que le *mot*, ce niveau reste encore une étape intermédiaire dans l'extraction des attributs.

Une heuristique triviale pour extraire les *mots* consiste à définir un ensemble de caractères séparateurs - généralement des espaces et des signes de ponctuation, puis parcourir le texte et retenir toute séquence de *caractères non séparateurs* comprise entre deux *caractères séparateurs* [183] [182]. Certains caractères peuvent être à la fois des caractères séparateurs et faire partie des formes (e.g. *' - . , :*). D'autres peuvent être soit des séparateurs, soit être des formes à part entière (e.g. *( ) < > ! ? =*). L'opération de segmentation n'est pas triviale et doit tenir compte des caractères qui entourent chaque caractère séparateur.

Les attributs extraits de cette façon s'appellent *formes graphiques* (ou simplement *formes*) et ne correspondent pas nécessairement à des *mots* dans le sens linguistique. Des nombres (e.g., *1.234.567,89*), des numéros de téléphone, des sigles (*S.N.C.F.*) sont des exemples de *formes* qui ne sont pas des *mots*. Pour cette raison, il est préférable d'utiliser le mot *forme* pour désigner l'unité de segmentation.

Les *occurrences* sont des instances d'une même *forme* à des endroits différents du texte.

Dans les applications de classement de spam il est souvent utile de convertir systématiquement certaines séquences courantes telles un numéro de téléphone ou un montant en une représentation de format : *99.99.99.99.99* (au lieu de *01.40.51.90.00*) ou *U$ 999,999.00* (au lieu de *U$ 110,865.22*). Ces transformations visent, certes, à réduire le nombre d'entrées du vocabulaire et surtout à retenir que le terme en question fait référence à un numéro de téléphone, ou à un montant en dollars plutôt que à sa valeur précise (il s'agit d'un *méta-attribut*, comme nous verrons par la suite). Une autre transformation souvent utile est la conversion systématique vers une casse de caractères unique, en général les minuscules.

D'autres traitements peuvent être utiles pour des applications plus pointues : le remplacement de certains mots par un un synonyme unique ou la levée d'ambiguïté pour des mots pouvant avoir

[3] Dans la bibliographie en langue anglaise l'opération de *segmentation* est appelée *tokenization*

des sens différents selon le contexte. En général, ces traitements, assez lourds, ne se justifient pas pour le classement de spam [59].

4.3.2 Niveau formes multiples

La perte d'information due à l'hypothèse d'indépendance statistique des formes n'est pas le seul inconvénient de la représentation de niveau formes simples. Les expressions et mots composés (e.g. *Recherche Documentaire*, *Apprentissage Artificiel*) perdent leur sens lorsque les mots sont considérés séparément.

Pour tenir compte de cet aspect, on utilise des termes qui sont constitués non pas d'un seul mot, mais d'une suite de mots. Les deux approches pour constituer des *termes* d'ordre supérieur sont [231] [138] :

Approche Syntaxique : cette approche consiste à considérer des *séquences de mots* ou même des *phrases* entières, conformes à la grammaire de la langue, comme *termes* [170] [94]. Dans ce cas, les termes sont significatifs, du point de vue sémantique ou syntaxique et n'ont pas forcément une taille fixe (en nombre de mots ou formes). Cette approche est dépendante de la langue et, à notre connaissance, aucun résultat de recherche utilisant cette approche pour le classement de spam n'a été publié.

Approche Statistique - cette approche consiste à utiliser un nombre fixe de *formes* (2, 3, ...) pour constituer les *termes* (des *n-grams* de niveau mot) [42]. Par exemple, dans un modèle *2-gram* la phrase *"Allons enfants de la patrie"* sera représentée par les termes {*"Allons+enfants", "enfants+de", "de+la", "la+patrie"*}. C'est toujours le modèle *BOW* qui est utilisé, mais l'utilisation d'une fenêtre glissante produit une prise en compte partielle de la dépendance statistique entre les mots. Il existe une équivalence implicite entre ce modèle de génération de messages et un processus de Markov, où l'état du processus est défini les $n-1$ derniers termes, mais les valeurs de probabilité visibles et utilisées sont les probabilités des états et non pas les probabilités de transition.

Plusieurs travaux [170] [94] [42] ont démontré que l'utilisation de représentations plus complexes n'améliore pas de façon significative l'efficacité de classement et parfois peut même la dégrader. La raison est que les termes d'ordre supérieur au mot sont plus significatifs du point de vue sémantique, mais moins du point de vue statistique : d'une part leur fréquence est plus faible puisque le vocabulaire est plus important, et aussi le nombre d'attributs croit exponentiellement avec l'ordre de la représentation [183] [231] [59], ce qui implique une utilisation d'un nombre d'échantillons plus important et un classificateur avec une inertie importante, inconvénient majeur dans le cas d'un processus évolutif tel le classement de spams.

Joachims [138] conjecture que le mot est la plus petite forme avec du sens sémantique, le point où se croisent la sémantique, la syntaxe et la morphologie, ce qui fait que ce niveau de segmentation est souvent, optimal pour un grand nombre de tâches.

Si l'utilisation des termes d'ordre supérieur au mot n'est pas toujours justifiée, le mélange de *phrases* ou *n-grams* avec des *mots* (e.g. mots + bi-mots) permet d'améliorer la qualité de classement [42] [231] [59].

Une variante de cette approche est le *OSB* (Orthogonal Sparse Bigrams) [242], utilisée par le filtre anti-spam CRM114[4] . Cette représentation permet d'intégrer l'interaction des mots non adjacents. Pour l'extraire les termes, il suffit de parcourir le texte avec une fenêtre de taille N (typiquement 5) et noter tous les couples de deux mots à l'intérieur, pour chaque position de la fenêtre. Le nombre de termes extraits pour un texte de longueur M est de l'ordre de $M*(N-1)$. Ainsi, la phrase *"Allons enfants de la patrie*" génère les termes {*"Allons enfants * * *", "Allons * de * *", "Allons * * la *", "Allons * * * patrie", "enfants de * *"*, ...}. Cette représentation semble astucieuse, mais elle présente les mêmes avantages et inconvénients des autres méthodes de représentation de niveau supérieur au mot.

[4] *CRM114* - http ://crm114.sourceforge.net

4.3.3 Niveau sub-mots

Dans ce niveau, deux types de représentations ont été expérimentées pour le classement des spams : les *n-grams* de niveau caractère et les *arbres de suffixes*.

La représentation par n-grams est équivalente, au niveau *caractères*, à l'approche statistique du niveau *mots multiples* : il s'agit de parcourir le texte avec une fenêtre glissante de taille n caractères. Ainsi, la phrase *"Allons enfants de la patrie"* serait représentée par les 4-grams {*"allo", "llon", "lons", "ons_", "ns_e", "s_en", "_enf", "enfa", "nfan", ...*}. Dans ce niveau de représentation, la génération de texte est modélisée par un processus de Markov d'ordre *n-1* [218] ce qui permet, comme pour le niveau des mots multiples, de restaurer partiellement la dépendance entre les parties successives du texte.

La modélisation de niveau caractère du langage naturel remonte à Shannon [238] pour l'évaluation de l'entropie asymptotique de la langue anglaise et la sécurité des systèmes cryptographiques [237]. Shannon modélisait la génération d'un texte par des processus de Markov d'ordre croissant, partant d'un processus sans mémoire. Le résultat de ses travaux ont été mis à jour plusieurs années après, par Cover [72] et Brown [39]

Si bien que largement utilisée depuis longtemps dans les applications commerciales de reconnaissance vocale [218], l'application au langage naturel écrit est resté restreint à la cryptanalyse [153], à la reconnaissance optique de caractères et à la détection et correction de fautes dans les textes [247], probablement à cause du succès limité de certaines expérimentations et de la représentation faite plus naturellement en mots que en *n-grams*.

À la fin des années 70, Suen [247] étudie les caractéristiques statistiques des n-grams et suggère son utilisation dans les applications de traitement de documents textuels pour des tâches plus complexes que la détection et correction de fautes typographiques. Dans les années 80 et 90, quelques travaux en recherche documentaire ont été publiés avec des résultats modérés [120, p. 116] [45] (identification d'adresses postales) ou alors non validés sur des corpus de taille suffisamment importante [120, p. 116] [83].

Cavnar [44] a utilisé des n-grams pour classer des articles de groupes de "news" *Usenet newsgroups* avec précision de 99.8 % pour un classement selon la langue et de 80 % pour un classement thématique. À la fin des années 90 des résultats plus robustes ont commencé à paraître pour des applications de classement [193], indexation [184] ou traduction automatique [89] de textes.

La représentation de texte avec des *n-grams* a plusieurs intérêts [138] [59] : la robustesse dans un environnement bruité, la capacité de identifier la similarité entre mots et/ou motifs, la capacité d'identifier et corriger les fautes d'orthographe et les déviations morphologiques, ainsi que des interactions intra et inter mots. La sensibilité au bruit est une caractéristique essentielle pour le classement des spams, à cause des fautes intentionnelles courantes (modélisées comme étant du bruit) visant à tromper les classificateurs.

La représentation par *n-grams* pour le filtrage de spams a été experimentée par Brien et Vogel [195], Keselj [144], Cormack [58], Sculley [230], Kanaris [141], Berger [19]. Lors de la dernière conférence TREC-2007, les deux classificateurs les plus performants utilisaient des *n-grams* pour la représentation des messages : Cormack [58] avec un classificateur à régression logistique [115] et Sculley avec une classificateur SVM [229]. Tous les deux utilisaient les 3500 premiers caractères du message brut.

Pampapathi [199] a utilisé des arbres de suffixes pour représenter les messages. Un arbre de suffixes est une structure hiérarchisée contenant tous les suffixes d'une chaine [186] [254] [192] [122]. Ces structures sont beaucoup utilisées pour l'indexation intégrale de texte et pour la recherche exacte de séquences similaires dans les génomes (bioinformatique). Ces structures ont des inconvénients mais des travaux récents proposent des solutions pour les minimiser : la place mémoire occupée [1] et la sensibilité au bruit (puisque la recherche est exacte) [271].

Les résultats de Pampapathi [199] semblent moins bons que ceux de Cormack et Sculley [58] [230] mais ils ne peuvent malheureusement pas être comparés : les corpus de messages sont différents, contiennent très peu de messages et, surtout, l'auteur a utilisé un protocole de validation croisée (*K-fold cross validation*) pour évaluer l'efficacité du classificateur - ce protocole n'étant pas adapté au classement en ligne, les résultats sont trop optimistes.

Dans les différentes évaluations, en particulier TREC 2007, les filtres les plus performants ont utilisé les *tétragrammes* de niveau caractère comme attribut. La raison, selon Gordon Cormack [59], est que, du fait que ce type d'attribut contient moins d'informations linguistiques que son équivalent de niveau mot, l'évidence suggère qu'il est capable de mieux capturer les interactions inter et intra mots et sont assez robustes vis-à-vis des fautes d'orthographe et des variantes morphologiques, des qualités qui semblent intéressantes dans un classificateur de messages électroniques.

4.3.4 Niveaux sémantique et pragmatique

Ces deux niveaux visent la compréhension du texte, des niveaux atteint lorsque le texte est lu par un humain et suffisant pour le classement des messages. Néanmoins, l'état actuel des techniques ne permet pas d'extraire automatiquement la sémantique d'un texte non structuré et de le représenter sous une forme opérationnelle [138, p. 16]. Il n'y a pas d'évidence montrant que ce niveau apporte une amélioration au classement de spams.

Certaines méthodes de classement de spam cherchent à identifier l'objet du message par la recherche de certaines expressions régulières capables de caractériser le sujet. Mais dans ces cas, il ne s'agit pas d'une méthode inductive, c'est-à-dire, on ne part pas du texte pour déduire son sens, mais on se contente de rechercher certains critères particuliers permettant de classer le message dans la catégorie spam.

4.4 Méta-attributs et termes synthétiques

Sahami et al [220] ont remarqué que l'inclusion d'attributs synthétiques (*domain features*) parmi les attributs textuels leur permettait d'améliorer l'efficacité de filtrage. Ils ont défini 35 attributs de ce type : des bouts de phrases du type *"be over 21"*, *"Only $"*, adresse de l'expéditeur finissant par *".edu"* (université américaine), présence de fichiers attachés, ...

On peut, par exemple, préfixer le terme pour préciser l'endroit où apparaît dans le message : *"body :money"*, *"subject :casino"* ou *"html :href"*.

Au contraire des attributs linguistiques, ces attributs sont, en général, définis à l'avance. On peut inclure n'importe lequel type de terme synthétique dans l'opération de classement. On peut, par ailleurs, voir les 800 règles de *SpamAssassin* comme un ensemble d'attributs synthétiques [59].

4.5 Contenu textuel ou méta-informations ?

Un message électronique contient, en plus de l'information textuelle - objet principal du message, des informations structurées tels les champs de l'en-tête et les méta informations MIME[5] , et peut aussi contenir des documents attachés de tout genre tels des images, documents bureautiques, pages HTML, ...

Depuis quelques années, on a vu apparaître des spams donc l'information textuelle est insérée dans des images, parfois bruités (*e.g.* figure 4.1), ou dans des documents bureautiques (*e.g. pdf*, *doc*, *xls*). De nombreuses solutions ont été proposées pour identifier ces types particuliers de messages [12] [109] [91] [259] [41] [21] [194]. Néanmoins, leur identification reste une opération spécifique (puisque applicable à ces seuls types de spam) et coûteuse, alors que ces spams sont souvent identifiés par les éléments textuels du message.

Zhang et al [274] et Martins et Cormack [81] ont comparé l'efficacité de classement lorsque on utilise seulement les en-têtes, seulement le corps des messages et quand on utilise le tout et ont remarqué des meilleurs résultats, en ordre décroissante, avec un mélange des en-têtes et le corps, les en-têtes seuls puis finalement le corps du message seul. Zhang a utilisé un classificateur SVM, tandis que Martins et Cormack ont utilisé un classificateur à régression logistique.

[5] Les méta informations MIME servent à indiquer comment le message est structuré, le codage des caractères, les fichiers attachés, ...

FIG. 4.1: Spam image

Lors de TREC 2007, Cormack [58] et Sculley [229] ont proposé de tronquer les messages et de ne prendre que les premiers N (autour de 3000) caractères, mais en leur totalité, sans aucune opération de sélection d'attributs.

Cette heuristique revient à, d'une part, retenir les en-têtes (avec un poids considérable) plus une partie initiale du message (celle où, en général, se trouve la partie textuelle) et d'autre part, fixer la quantité d'information utilisée, éliminant ainsi le besoin de normalisation par rapport à la taille du message.

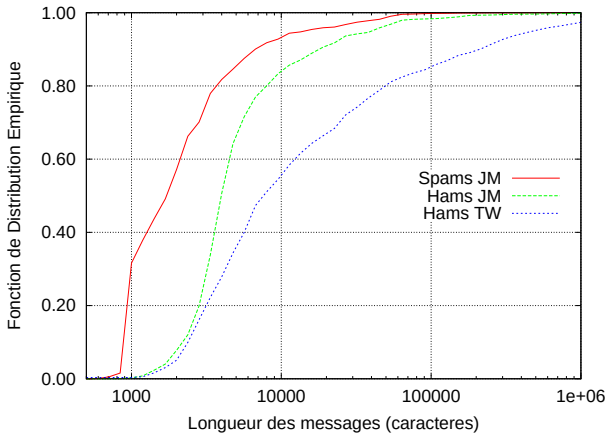


FIG. 4.2: Distribution des longueurs des messages - spam et ham de JM et ham de TW. Les différences de profils entre JM et TW justifient les différences dans les longueurs des messages légitimes : JM est abonné à plusieurs listes de discussion (messages courts) tandis que TW échange souvent des messages avec des documents de travail en attaché.

La longueur des en-têtes des messages est concentrée entre 1000 et 3000 caractères et dépend de la classe (voir Tab. 4.1). La différence entre les classes peut s'expliquer par :

– Les spams ont, en général, un nombre inférieur d'en-têtes *"Received : from"*, puisqu'ils sont, pour la plupart, envoyés directement par des *zombies*[6], au serveur de messagerie du

[6]Zombies - une machine zombie est un ordinateur contrôlé à l'insu de son utilisateur par un pirate informatique. Ce dernier l'utilise alors le plus souvent à des fins malveillantes, par exemple afin d'attaquer d'autres machines en dissimulant sa véritable identité (`http://fr.wikipedia.org/wiki/Machine_zombie`).

destinataire, alors que les messages légitimes passent souvent, au moins, par un serveur de soumission de messages (serveur sortant).
- Les spams sont, en général, construits de façon minimale avec juste les en-têtes nécessaires pour qu'ils puissent sembler légitimes.
- Les serveurs de listes de diffusion (y compris les *groupes de discussion*, *newsletters*, ...) ajoutent un nombre assez important d'en-têtes (donc, une partie en-tête plus longue). La différence est plus importante lorsque le destinataire est abonné à beaucoup de listes. En général, ce type de message concerne plutôt des messages légitimes.
- Le récent déploiement de techniques de signature de messages, telles *DKIM*, fait que la longueur de la partie en-tête a considérablement augmenté pour les messages légitimes. Mais il s'agit d'une situation transitoire - il est possible que les spams soient aussi affectés dans un futur un peu plus lointain par cette technologie.

Aussi, selon le nombre de serveurs de messagerie traversés en interne dans l'organisation du destinataire, chacun de ces serveurs ajoutant au moins un nouveau en-tête, la longueur totale de la partie en-tête n'est pas la même pour des destinataires dans des organisations différentes. Il est légitime de penser que cette longueur optimale (autour de 3000 caractères) dépend des destinataires des messages et peut évoluer dans le temps. Et d'autre part, la structuration générale des messages (hams et spams) peut aussi évoluer dans le temps, et cette quantité peut ne pas être optimale.

Comme nous verrons plus loin, le classificateur discriminant linéaire utilisé dans nos expérimentations utilise une variante de cette heuristique qui consiste à retenir la totalité des en-têtes, toute méta-information contenue dans le corps du message plus une quantité fixe (512 caractères) de chaque partie du corps du message.

	minimum	maximum	médiane	moyenne
message	667	890348	2033	5213
en-têtes	503	6634	971	990
corps	18	889073	1097	4223

(a) Spams JM - 25314 messages

	minimum	maximum	médiane	moyenne
message	949	5209194	4732	17834
en-têtes	741	7086	2851	2641
corps	32	5207142	1568	15192

(b) Hams JM - 4167 messages

	minimum	maximum	médiane	moyenne
message	1074	9929961	9658	145339
en-têtes	885	6087	1799	1934
corps	2	9927662	7581	143404

(c) Hams TW - 3452 messages

TAB. 4.1: Distribution de la longueur des messages

En fait, les informations contenues dans le corps et dans les en-têtes des messages sont de nature différentes. La partie initiale du corps contient, généralement, le message textuel (dans une langue tel l'anglais ou le français), tandis que les en-têtes contiennent des informations structurées

indiquant la façon dont le message a été constitué ainsi que son parcours depuis sa création (la séquence de serveurs de messagerie traversés par le message). Cette partie structurée permet de attribuer une sorte de "*réputation*" au message.

4.6 Sélection des attributs - réduction de la dimension

Dans les applications de classement de documents textuels, il n'est pas rare que la dimension (nombre d'attributs différents extraits des exemples) atteigne 10^4 à 10^7, dont très peu pertinents [25] [171]. L'utilisation de termes d'ordre supérieur au mot contribue à cet explosion.

Le premier problème lié à la dimension est la complexité algorithmique (temps de traitement et place de stockage requise). Idéalement, on privilégie des algorithmes dont la complexité augmente au plus linéairement avec la taille du problème. Des algorithmes faisant appel à, par exemple, des inversions de matrices, peuvent devenir inutilisables avec des dimensions trop importantes.

Intuitivement, la vitesse d'apprentissage décroit avec l'augmentation du nombre d'attributs [25]. Langley et Iba [161] [162] ont démontré analytiquement que pour un classificateur k-NN, le nombre d'exemples nécessaires augmente, en moyenne, exponentiellement avec le nombre d'attributs.

Le sur-ajustement (*overfitting*) est une autre conséquence du nombre important d'attributs [23, p. 9], [126, p. 194], [138, p. 16]. Il s'agit d'attribuer à un problème une complexité (dimension) plus importante qu'il ne le faut, ce qui diminue la capacité de généralisation du classificateur.

Cette phase de *sélection d'attributs* ou *réduction de la dimension* de la représentation est effectuée, après la segmentation.

Pour ce faire, il y a deux approches, selon la méthode utilisée [138, p. 16] [231] : sélection d'un sous-ensemble d'attributs ou extraction des attributs.

4.6.1 Sélection de sous-ensembles d'attributs

Il s'agit de sélectionner et supprimer les attributs non pertinents ou qui n'ont pas d'influence dans le classement. Il y a encore deux sous-approches : *filtrage* et *wrapper*.

Approche *wrapper*

Le principe de cette approche est la recherche de l'ensemble d'attributs qui minimise l'erreur de généralisation du classificateur. Dans une première méthode, on va chercher à construire l'ensemble des attributs de façon incrémentale, les ajoutant un par un, à partir d'un ensemble vide [231], [5, p.106]. À chaque étape, les attributs ajoutés sont ceux qui contribuent le plus à la réduction de l'erreur de classement. L'opération s'arrête lorsque l'erreur de classement ne diminue plus de façon significative. C'est l'approche de *sélection directe*[7]. Une variante est la *sélection inverse*[8] : il s'agit de partir plutôt de l'ensemble complet et d'enlever les attributs, un par un, jusqu'à ce que l'erreur de classement cesse de diminuer.

L'intérêt de cette approche est son indépendance de l'algorithme de classement, vu comme une boîte noire. Par contre, l'inconvénient est la complexité algorithmique : il faut évaluer, à chaque pas, l'erreur de classement marginal pour chaque attribut non encore traité. Une diminution de la complexité peut être obtenue dans les cas où il est possible d'éliminer plusieurs attributs, à chaque pas de traitement. Néanmoins, vu l'ordre de grandeur des dimensions en jeu, l'approche devient intraitable pour les problèmes de classement de textes. Pour cette raison, cette approche est très rarement utilisée dans ces problèmes [231].

Dans le cas particulier du classement de spams, s'agissant d'un processus évolutif (non stationnaire), on cherchera à faire, tant que possible, de l'apprentissage incrémental, ce qui nécessite la mise à jour fréquente du modèle et rend cette approche encore moins intéressante.

[7] En anglais, *forward selection*
[8] En anglais, *backward selection*

Approche filtrage

Dans cette approche, la pertinence des termes est évaluée uniquement par rapport à des caractéristiques des attributs, sans tenir compte de l'erreur de classement et, par conséquent, du type d'algorithme de classement.

Il s'agit d'évaluer l'importance de l'attribut et de sélectionner les plus pertinents. Sebastiani [231] cite quelques critères courants pour les tâches de classement de textes. La *pertinence* d'un attribut, par rapport au classement s'interprète comme une *dépendance statistique* entre la classe et l'attribut - c'est ce qui essayent d'évaluer, la plupart de ces critères.

- **Élimination des formes banales**[9] - il s'agit de supprimer les termes qui ont un faible pouvoir discriminant. Par exemple, les articles (*le*, *la*, *les*, ...). Il s'agit d'une méthode dépendante de la langue.
- **Fréquence dans les Documents - DF**[10] - Il s'agit du nombre de documents dans lesquels l'attribut est présent. Ce critère sert juste à éliminer les termes peu fréquents, statistiquement peu représentatifs. Il n'est pas rare que certains filtres éliminent aussi les termes dont la fréquence est élevée dans toutes les classes et qui finissent par être peu discriminants.
- **Gain d'Information - IG**[11] - Il s'agit d'une mesure issue de la théorie d'information évaluant la diminution d'incertitude moyenne de la classe lorsque l'on tient compte de la présence/absence du terme t_i [73, p. 19].

$$IG(t_i) = H(C) - H(C|t_i)$$

$$IG(t_i) = -\sum_{c \in C} P(c) \log_2 P(c) + \sum_{\substack{c \in C \\ t \in T_i}} P(t) P(c|t) \log_2 P(c|t)$$

où $C = \{ham, spam\}$, représente l'ensemble des classes, et $T_i = \{t_i, \bar{t_i}\}$ indique la présence ou absence du terme dans le document.

> **NOTE** : En fait, ce critère correspond à l'*Information Mutuelle*, tel qu'il est connu en Théorie d'Information, entre le terme t_i et la classe, mais il est appelé autrement pour ne pas être confondu avec le critère *Rapport d'Association*, connu, *à tort*, par l'appellation *Information Mutuelle*. Il y a parfois confusion et l'on trouve des publications utilisant l'un des critères comme étant l'autre - voir par exemple Battiti [15] ou Zaffalon [272], qui ont utilisé ce critère, tout en le désignant, *à juste titre*, "*Mutual Information*".

- **Information Mutuelle - MI**[12] - Ce critère a été proposé par Church et Hanks [52] sous le nom *Rapport d'Associaton*[13], un rapport dérivé de l'*information mutuelle*, et défini comme étant :

$$I(t,c) \; = \; \log_2 \frac{P(t,c)}{P(t)\,P(c)} \; = \; \log_2 \frac{P(t|c)}{P(t)} \; = \; \log_2 \frac{P(c|t)}{P(c)}$$

Cette valeur peut être estimée à partir des valeurs de la table de contingence de l'attribut versus la classe, ce qui permet d'estimer les quatre valeurs correspondantes aux quatre cases de la table. Deux valeurs dérivées sont la moyenne et le maximum :

$$I_{avg}(t) = \sum_{c \in \{ham, spam\}} P(c)\, I(c,t)$$

$$I_{max}(t) = \max_{c \in \{ham, spam\}} I(c,t)$$

[9] Les formes banales sont aussi connues sous la dénomination *termes-outils*. En anglais : *stop word removal*
[10] En anglais, *Document Frequency*
[11] En anglais, *Information Gain*
[12] En anglais, *Mutual Information*
[13] En anglais, *Association Ratio*

Le problème avec ce critère est qu'il est fortement dépendant de la probabilité marginale des termes [266], ce qui fait que, pour des termes avec probabilités conditionnelles $P(c|t_i)$ équivalentes les termes rares auront des scores plus importants que des termes courants. Ces scores ne sont donc pas comparables si la plage de variation des probabilités à priori des termes est trop importante.

– **Test de χ^2** - Le test statistique de χ^2 permet d'évaluer l'indépendance statistique du terme t et de la classe c, à partir de la table de contingence du terme versus la catégorie.

$$\chi^2(t_k, c_i) = \frac{|T|\,(P(t_k, c_i)\,P(\bar{t_k}, \bar{c_i}) \;-\; P(t_k, \bar{c_i})\,P(\bar{t_k}, c_i))^2}{P(t_k, c_i)\,P(t_k, \bar{c_i})\,P(\bar{t_k}, c_i)\,P(\bar{t_k}, \bar{c_i})}$$

Cette valeur peut être comparée à une distribution de χ^2 avec un degré de liberté pour tester l'indépendance entre le terme et la classe.

Yang et Pedersen [266] et Rogati et Yang [215] ont comparé, pour une application de classement thématique, l'efficacité de cinq critères, dont certains cités ci-dessus et ont trouvé des meilleurs résultats pour χ^2, IG et DF, devant MI. Yang et Pedersen observent une très forte corrélation entre ces trois critères et suggèrent qu'ils peuvent être choisis indifféremment, selon les caractéristiques de l'application. Dans ces deux études, l'efficacité était encore optimale pour des ensembles d'attributs de dimension inférieure à 10 % de la dimension d'origine. Les deux études ont utilisé le même corpus de documents : *REUTERS*.

D'autres critères sont cités par Sebastiani [231] et ses références.

4.6.2 Extraction (ou construction) d'attributs

Le but des méthodes d'extraction d'attributs est, non pas de supprimer les attributs inutiles mais, d'en créer des nouveaux, synthétiques, représentant aussi bien, du point de vue de l'algorithme de classement, le document de départ, mais avec moins d'attributs.

Les méthodes les plus courantes dans les applications de classement textuel sont des traitements morphologiques des termes : il s'agit d'effectuer des traitements au niveau de la forme des termes (mots), normalement pour regrouper les attributs "équivalents".

Lemmatisation - c'est le regroupement sous une forme canonique (en général à partir d'un dictionnaire) des occurrences du texte [164]. En français, ce regroupement se pratique en général de la manière suivante :
- les formes verbales à l'infinitif
- les substantifs au singulier
- les adjectifs au masculin singulier
- les formes élidées à la forme sans élision

Par exemple, les mots "*petit*", "*petite*", "*petits*", "*petitesse*" sont tous convertis en "*petit*". Aussi, les différentes formes du verbe "*être*" : "*suis*", "*est*", "*sommes*", ... sont convertis à l'infinitif.

Extraction des racines : Stemming - c'est une transformation similaire à la lemmatisation, mais il s'agit juste d'extraire les racines des termes, par suppression des affixes. En langue anglaise, l'algorithme de stemming le plus courant est celui proposé par Porter [182, p.31] [203].

Ces deux méthodes sont parfois incluses dans la catégorie "séléction d'attributs", et sont dépendantes de la langue. Dans un environnement multi-langues, l'implémentation des heuristiques spécifiques à chaque langue est nécessaire.

Leopold et Kindermann [169], Lewis [173] ont montré que, dans le cas particulier des algorithmes de classement du type SVM, des méthodes de sélection d'attributs du type *lemmatisation* et *stemming* n'améliorent pas leur efficacité. appliqués à des langues telles le français et allemand.

D'autres méthodes d'extractions/construction d'attributs existent, telles la PCA (Analyse des Composantes Principales) [23, p. 561], ces méthodes restent coûteuses et ne sont pas utilisées dans les applications de classement textuel.

4.6.3 Réduction de dimension dans les applications de filtrage de spam

Dans les paragraphes précédents nous avons présenté quelques méthodes courantes de sélection d'attributs utilisées dans les applications de traitement et classement de documents textuels. Ces méthodes sont toutefois peu ou pas utilisées dans les applications courantes de filtrage de spam, en particulier dans les filtres bayésien naïfs. La raison première est le besoin d'une opération qui relève le plus souvent d'une optimisation, donc complexe à mettre en oeuvre et de coût de traitement souvent élevé. Le flot de messages n'étant pas un processus stationnaire les classificateurs doivent être mis à jour en permanence. Cela fait que l'on privilégie les méthodes plus simples de sélection d'attributs.

Ces méthodes visent, en réalité, à réduire la dimension du vocabulaire construit lors de l'apprentissage. L'approche que l'on trouve fréquemment dans les classificateurs *"dit bayésiens naïfs"* issus de la communauté des logiciels libres est plutôt de garder la taille de départ du vocabulaire et de sélectionner, dans le message à classer, les N termes les plus "informatifs"[14]. Le critère définissant la relation d'ordre est la quantité d'information sur la classe contenue dans chaque terme :

$$\begin{aligned} I(t) &= 1 - H(P_t(c = spam)), \quad t \in \mathcal{D} \\ H(x) &= x \log \frac{1}{x} + (1 - x) \log \frac{1}{1 - x} \end{aligned} \tag{4.3}$$

La fonction $I(t)$ est convexe, atteint sa valeur minimale pour $P = 0.5$ et est symétrique par rapport au minimum. L'utilisation d'une autre fonction quelconque avec ces caractéristiques comme critère permet de préserver la relation d'ordre. Dans la pratique, la fonction utilisée est plutôt :

$$I'(t) = |P_t(c = spam)) - 0.5| \tag{4.4}$$

Pour le problème de filtrage de spam, Drucker et al [92] ont expérimenté plusieurs critères de sélection d'attributs, en particulier ceux qui avaient déjà été comparés par Yang et Pedersen [266] et ont estimé que la suppression des formes banales dégrade l'efficacité et que pour les SVM, l'absence de sélection d'attributs ne modifie pas ni l'efficacité ni les performances de l'algorithme de classement. Aussi, les algorithmes de "boosting" utilisant des arbres de décision choisissent naturellement les meilleurs attributs comme partie intégrante de l'apprentissage [92].

Dans l'état actuel des recherches, l'utilité d'une méthode générale de réduction du nombre d'attributs reste un problème ouvert.

Pour le problème de classement thématique de textes, certains travaux présentent des résultats, en apparence, contradictoires. Par exemple, pour le classement thématique utilisant des SVMs, Drucker et Vapnik [92] Leopold et Kindermann [169] Bekermann [17] et Lewis et Yang [173] ont estimé que la sélection d'attributs n'était pas utile, alors que Rogati et Yung [215] Gabrilovich et Milkovitch [111] estiment le contraire. Ces auteurs avancent des explications, tout à fait plausibles, telles les caractéristiques statistiques du corpus (vocabulaire important et diversifié) et les caractéristiques particulières des applications [111] [17], la robustesse des SVMs aux attributs non pertinents [173].

Aussi, il ne semble pas raisonnable d'étudier des méthodes de sélection d'attributs indépendamment de l'algorithme de classement et des données [59].

4.7 Normalisation de la taille des documents

Dans les applications de recherche documentaire, se basant sur une étude des collections de documents de TREC et sur des résultats vraisemblablement incohérents obtenus avec ces collections, Singhal et al [243] ont remarqué que des documents plus longs ont plus de chances de satisfaire les critères d'une requête que les documents courts, et que l'inclusion de la longueur

[14] Dans les logiciels libres de filtrage de spam, le nombre de termes varie, typiquement, entre 15 et 100, selon le type de terme

dans l'évaluation de la pertinence des documents donnait des résultats plus satisfaisants. L'idée de cette inclusion est de ramener les caractéristiques des documents à celles d'un document de taille fixe normalisée.

Dans les applications de filtrage de spam, la plage de variation de la longueur des messages est assez large : de quelques kilo-octets à quelques méga-octets et les spams sont souvent bien plus courts que les hams, rarement dépassant 20 Ko.

Dans les applications de filtrage de spam, les approches les plus courantes de normalisation de la longueur sont :

1. **Utiliser les N termes les plus pertinents** - C'est l'approche proposée par Graham [116] (15 termes), et souvent utilisée dans les applications distribuées sous licence GPL. Il s'agit d'extraire les termes trouvés dans le message entier et de ne retenir que les plus pertinents. Cela fait que les documents sont vus comme ayant toujours une taille fixe. Cette méthode sert à la fois pour normaliser la longueur du document et à réduire la dimension du problème.
2. **Tronquer le message à une longueur fixe** Cormack [58] a utilisé des n-grams pour représenter des messages tronqués à environ 3500 caractères et observé des meilleurs résultats que si la totalité du message avait été retenue. Aussi, l'utilisation de n-grams de taille fixe fait que le nombre d'attributs est fixe comme la longueur du document.
3. **Utiliser la norme Euclidienne** (Norme L2), pour normaliser les vecteurs d'attributs [226].

$$x_{i-normalized} = \frac{x_i}{\sqrt{< x_i, x_i >}}$$

   Bien entendu, cette méthode ne peut pas être appliquée que pour certains types d'algorithmes de classement et pour certaines représentations.
4. Ne rien faire.

4.8 La Multiplicité des Langues

La multiplicité des langues dans les applications de classement de spams est un problème connu, mais à peine effleuré. L'hypothèse courante que la messagerie électronique est constituée de messages en langue anglaise ou utilisant uniquement l'alphabet latin n'est pas justifiable [59]. Néanmoins, la majorité des travaux publiés utilisent des corpus en langue anglaise et, assez souvent, ne le mentionnent même pas. Une explication probable est l'environnement de recherche : les travaux sont souvent effectués par des anglophones, ou utilisant des corpus de messages en anglais. Accessoirement, quelques travaux publiés sont plus spécifiques au traitement de messages en d'autres langues, mais nous n'avons pas trouvé, à ce jour, des publications concernant la problématique de la multiplicité linguistique dans les spams.

Du côté des messages légitimes, dans les pays non-anglophones, il n'est par rare qu'une personne puisse avoir des correspondants dans plusieurs pays et écrire et recevoir ses messages dans les langues de ses différents correspondants. Si la langue dominante aux États Unis est l'anglais, en France, le contenu d'une boîte aux lettres peut varier du tout en français à un mélange de français, anglais, allemand, italien, ... avec des proportions variables selon l'utilisateur. Il peut aussi arriver que le même message puisse contenir des parties en plusieurs langues.

Du côte des spams, MessageLabs [130] rapporte que aux États Unis la langue dominante dans les spams est l'anglais pour plus de 90 %, tandis que ce chiffre peut baisser à de l'ordre de 50 % dans d'autres pays. En même temps, il n'est pas rare de trouver des spams dont le contenu est un mélange de plusieurs langues. Il n'est pas raisonnable de considérer comme vraie l'hypothèse que les spams sont des messages construits de façon cohérente.

Du point de vue linguistique, les langues se classent selon leur morphologie en : agglutinante, flexionnelle, isolante, synthétique ou polysynthétique [28, p. 41] et [97]. Ce classement indique,

par exemple, comment des nouveaux mots peuvent se créer à partir des morphèmes[15]. D'autre part, ce classement se fait selon les caractéristiques dominantes de la langue : chaque langue présente, généralement, des caractéristiques de plusieurs classes. Ces différences morphologiques font que les opérations classiques de réduction de dimension fondées sur la recherche des racines ou décomposition en morphèmes (telles la lemmatisation et stemming) sont très fortement dépendantes de chaque langue.

Des langues différentes peuvent impliquer aussi des codages de caractères différentes. Alors que le code ASCII sur 7 bits suffit pour coder la langue anglaise, d'autres langues utiliseront des jeux de caractères différents et même du codage sur plusieurs octets.

L'approche la plus courante dans les applications multi-langues[16] de recherche documentaire est la traduction de la requête soit dans les différentes langues des documents soit dans une langue pivot [120, p. 149-179] [146].

Dans les applications de catégorisation de textes, les approches les plus fréquentes sont l'extraction de ontologies [85], ce qui remonte l'analyse à un niveau conceptuel (sémantique), ou alors, une extension multi-langue de l'approche déjà utilisée pour les problèmes mono-langues [132] [24] [6] [208]. Dans cette dernière approche, on insère un niveau de traduction qui peut être avant ou après la segmentation du texte, avec génération d'un vocabulaire dans une langue cible ou dans toutes les langues possibles des documents à classer[17].

Jalam [132] et Biskri [24] suggèrent l'utilisation de n-grams de niveau caractère pour la segmentation vue la propriété intrinsèque d'extraction de la racine des termes (stemming) sans avoir à faire appel à des heuristiques dépendantes de la langue.

Osgur [198] a proposé une méthode de filtrage de spam, pour des langues agglutinantes[18], (en particulier la langue Turque). Sa méthode consiste à extraire la racine de chaque mot trouvé dans le texte, avant de la transmettre au classificateur. Ciltik [53] expérimenté des *n-grams* de niveau caractère (après traitement morphologique) et comparé les résultats de classement de messages en anglais et en turc. Ces expérimentations sont intéressantes puisque démontrent l'intérêt d'utilisation de *n-grams* de niveau caractère, déjà prévu par Jalam, néanmoins elles ont l'inconvénient d'avoir été faites dans un contexte mono-langue.

La prise en compte de l'aspect multi-langue dans le classement de spams présente quelques difficultés qui n'existent pas dans le classement de documents textuels courants.

1. Identifier la langue du message n'est pas une opération ni facile ni fiable. D'une part les spams sont des documents créés sans aucun souci de conformité (bien au contraire) et d'autre part, la longueur des textes n'est généralement pas suffisante pour détecter la langue. Dans la boîte aux lettres de l'auteur, au mois de novembre 2009, la longueur moyenne[19] des spams est de 5,3 Ko (dont 90 % ont une longueur inférieure à 5,7 Ko) et celle des hams est de 16,9 Ko (dont 90 % ont une longueur inférieure à 6,9 Ko).
2. On trouve souvent, dans les spams, des morceaux de texte en plusieurs langues dans le même message, ou alors des mots "synthétiques", ajoutés dans le but de tromper les méthodes de segmentation fondées sur le caractère régulier des messages.
3. Le vocabulaire utilisé par le classificateur est l'union des vocabulaires de toutes les langues présentes dans les boîtes aux lettres. Même si les langues sont souvent très différentes, il n'est pas raisonnable de considérer les vocabulaires comme étant disjoints (en particulier à cause des mots communs).
4. Les messages électroniques contiennent, en plus de la partie rédigée dans un langage naturel, des parties codifiées (p. ex. des URL) et des méta-informations telles les en-têtes ou des

[15] En linguistique, on définit un morphème, ou radical, comme la plus petite unité porteuse de sens qu'il soit possible d'isoler dans un énoncé.

[16] En anglais, *Cross Language Information Retrieval*

[17] On a trouvé des références à l'utilisation de l'*Espéranto* comme langue cible, mais aucune publication dédiée à ce cas particulier

[18] Dans une langue agglutinante les mots sont créés par ajout d'affixes (généralement des suffixes) à un radical. Le *basque*, le *turc*, le *finlandais*, le *japonais* et le *coréen* sont des exemples de langues agglutinantes.

[19] Ces valeurs incluent les en-têtes, les informations de mise en forme (HTML) et les éventuels fichiers attachés et images.

données de mise en forme en langage HTML. Dans les applications classiques de classement de textes, ces méta-informations ne sont pas pertinentes et sont souvent ignorées. Dans les applications de filtrage de spam, au contraire, elles sont souvent plus pertinentes que le contenu textuel lui-même [81]. Ceci ne concerne pas la problématique multi-langue du filtrage de spams, mais correspond à une "langue" de plus à tenir compte dans le classement.

A notre connaissance, aucune des solutions anti-spam disponibles ne tient compte ni de la langue ni du jeu de caractères non conventionnel parfois utilisé. Certaines se limitent à refuser un message juste parce que le jeu de caractères est celui d'un pays asiatique ou de l'est (ISO-2022-JP, Big5, Shift-JIS, Windows-1251, KOI8-r) et apparait souvent dans les spams. Mais ce genre de critère trivial n'est pas ce que l'on peut appeler un critère de filtrage de spam.

4.9 Conclusions

Dans ce chapitre nous avons fait le tour des aspects liés à représentation des messages, présentant les points qui sont particuliers à l'application de filtrage de spam. Il s'agit d'un aspect important puisque du choix d'une bonne représentation dépend l'efficacité de l'algorithme de classement. La représentation idéale est celle capable d'extraire et de mettre en valeur les caractéristiques les plus discriminants des messages à classer.

Dans la section 4.5 nous avons pu distinguer deux types de contenu dans un message : le contenu textuel proprement dit et les méta informations. Cette distinction est un point importante dans le contexte de cette thèse. D'une part, rien ne laisse supposer que leur pouvoir discriminant soit identique. D'autre part, on peut penser que, pour des destinataires différents, la diversité du contenu et du mode de distribution des spams sont moins importants que ceux des messages légitimes. On peut aussi penser que les méta-informations sont plus constantes dans un environnement multi-utilisateurs et multi-langues. Ainsi, le choix de l'endroit où prendre l'information ou la combinaison des deux types d'information semble être un aspect important dans la conception d'un le filtrage partagé dans une communauté.

Dans les différentes évaluations, en particulier TREC 2007, les filtres les plus performants ont utilisé les *tétragrammes* de niveau caractère comme attribut. La raison, selon Gordon Cormack [59], est que, du fait que ce type d'attribut contient moins d'informations linguistiques que son équivalent de niveau mot, l'évidence suggère qu'il est capable de mieux capturer les interactions inter et intra mots et sont assez robustes vis-à-vis des fautes d'orthographe et des variantes morphologiques, des qualités qui semblent intéressantes dans un classificateur de messages électroniques

Ces différentes évaluations semblent aussi démontrer que les opérations plus complexes de sélection d'attribut n'apportent pas d'amélioration significative ou alors se trouvent noyées dans l'incertitude de mesure.

Enfin, vue la diversité des facteurs - la complexité structurelle des messages, la multiplicité des langues, l'évolution des pratiques de composition, ... - certaines caractéristiques de la représentation idéale des messages, telles l'unité élémentaire de représentation ou le nombre d'attributs à prendre en compte, ne peuvent être que le résultat des expérimentations et sont susceptibles d'évoluer avec le temps.

CHAPITRE 5

L'Apprentissage Artificiel

> Of course it's very interesting to know how humans can learn. However, this is not necessarily the best way for creating an artificial learning machine. It has been noted that the study of birds flying was not very useful for constructing the airplane.
>
> *Vladimir N. Vapnik*

5.1 Introduction

L'apprentissage artificiel (ou apprentissage statistique) essaye de résoudre des problèmes tels :

> *Given some examples of complex signals and the correct decisions for them, make decisions automatically for a stream of future examples* [209].
>
> *Classer automatiquement un flot de messages arrivant sur un serveur de messagerie en messages légitimes et messages indésirables. Un ensemble de messages de chaque classe est donné, à titre d'exemple.*

Dans ces exemples, on distingue une application avec deux parties :

- **Apprentissage**[1] - c'est l'analyse des exemples pour déterminer quelle est la "fonction" capable d'exprimer, le mieux possible, la relation entre chaque exemple et une valeur associée (classe ou décision, dans les exemples ci-dessus). On associe à cette phase un algorithme d'apprentissage.
- **Estimation** - c'est l'application de la "fonction" trouvée lors de la phase d'apprentissage pour *estimer* ou *prévoir* la valeur (ou étiquette) à associer à des cas nouveaux. On associe à cette phase un algorithme d'estimation.

Formellement, un *algorithme d'apprentissage* reçoit, en entrée, un ensemble d'*exemples* $S^N = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_N}, y_N)\}$, $(\mathbf{x_i}, y_i) \in \mathcal{X} \times \mathcal{Y}$, et produit comme résultat une *hypothèse* $h : \mathcal{X} \mapsto \mathcal{Y}, h \in \mathcal{H}$. Les exemples sont des échantillons *i.i.d.* issus d'une distribution $P(\mathbf{X}, Y)$ inconnue.

Les éléments de $\mathcal{X}$ sont les représentations des *"objets que l'on étudie"*, sous la forme d'*attributs* (ou features), et ceux de $\mathcal{Y}$ sont les valeurs que l'on peut associer à chaque objet. La représentation des objets textuels, en particulier des messages électroniques, est traitée dans le chapitre 4.

Une *hypothèse* est la fonction "construite" par l'algorithme d'apprentissage. Cette fonction permettra, par la suite, grâce à un algorithme d'estimation, d'associer un élément de l'ensemble $\mathcal{Y}$ à élément de $\mathcal{X}$ absent dans l'ensemble des exemples d'apprentissage :

[1] Dans la bibliographie on trouve parfois les expressions *apprentissage* et *construction d'un classificateur* comme équivalentes.

$$\hat{y} = h(x)$$

Un espace d'hypothèses $\mathcal{H}$ est l'ensemble (ou famille) de fonctions qui contient toutes celles passibles d'être choisies par l'algorithme d'apprentissage.

La *généralisation* est la propriété d'un algorithme de pouvoir estimer correctement la valeur ou la classe à associer à des cas ne faisant partie des exemples d'apprentissage. Idéalement on aimerait que l'apprentissage puisse consommer le moins possible d'exemples. Un classificateur ayant besoin de balayer l'ensemble $\mathcal{X} \times \mathcal{Y}$ entier présente peu d'intérêt. Cette propriété peut dépendre des algorithmes mis en oeuvre mais aussi du problème à résoudre : certains problèmes sont plus *"durs"* que d'autres.

Il est important d'insister sur le fait que nous travaillons toujours sur des *représentations des objets* et non pas sur les objets eux mêmes. Bien entendu, des objets différents peuvent avoir la même représentation. Un choix judicieux des attributs permet parfois d'enlever toute ambiguïté de représentation mais cette discussion, malgré son intérêt, relève plutôt de l'*"ingénierie des attributs"*.

Régression et Classement

Lorsque l'ensemble $\mathcal{Y}$ est un ensemble ordonné, tel $\mathbb{R}$ ou $\mathbb{Z}$, il s'agit d'une application de ***régression***. Dans le cas contraire, un ensemble sans un ordre défini, comme par exemple $\{pomme, orange, raisin\}$ ou $\{sain, malade\}$, il s'agit d'une application de ***classement***.

Dans le problème qui nous concerne, le filtrage de messages électroniques, les étiquettes appartiennent à un ensemble de catégories (ou classes) $\mathcal{Y} = \{ham, spam\}$. Pour des raisons de simplification de notation et sauf besoin particulier, nous utiliserons, le plus souvent, l'ensemble binaire ordonné $\mathcal{Y} = \{0, 1\}$.

Il est courant, dans les problèmes de classement binaire, que le résultat de l'hypothèse ne soit pas une des classes mais une valeur réelle ou entière, un *score* indiquant, par exemple, la probabilité d'appartenir à une des classes - ce sont des *classificateurs souples* ("soft classifiers"). Dans le cas où le classificateur ne délivre que la catégorie associée, sans aucune autre indication, il s'agit d'un *classificateur rigide* ("hard classifier"). En pratique, les classificateurs rigides sont construits à partir d'un classificateur souple auquel on applique un seuillage.

5.2 Apprentissage Statistique

L'analyse statistique des algorithmes d'apprentissage considère, souvent, un contexte où les exemples résultent de tirages aléatoires indépendants et identiquement distribués d'un ensemble d'une distribution de probabilités en $\mathcal{X} \times \mathcal{Y}$ statique et inconnue [88]. Ainsi, un problème d'*apprentissage*, en informatique, est souvent traité comme un problème d'*estimation*, en statistique, utilisant des vocabulaires différents pour représenter des concepts équivalents [260, p. xi].

Formellement, le processus d'apprentissage peut être vu comme un processus de minimisation d'une fonctionnelle de coût [255, p. 17] [256, p. 20] [34]. Il s'agit de choisir dans un ensemble de fonctions possibles $\mathcal{H}$ celle satisfaisant le mieux possible un critère de qualité tel un **risque**, défini par l'espérance du coût :

$$R(h(\mathbf{x})) = \mathbf{E}_{\mathbf{x}}[C(\mathbf{x}, h(\mathbf{x}))] = \int C(\mathbf{x}, h(\mathbf{x})).dF(\mathbf{x}) \tag{5.1}$$

avec $\mathbf{x} \in \mathcal{X}$ et F, la distribution de probabilité de $\mathbf{x}$, est définie en $\mathcal{X}$ et intégrable pour tout $h \in \mathcal{H}$.

Le risque est nul lorsque l'hypothèse répond parfaitement à tous les objets qui lui sont présentés.

Le risque empirique

En pratique, la distribution F est inconnue mais on dispose d'un ensemble d'observations (exemples) $S^N = \{(\mathbf{x_1}, y_1), ..., (\mathbf{x_N}, y_N)\}$ indépendants et identiquement distribués de la distribution F. Aussi, sans perte de généralité, l'hypothèse choisie par le processus d'apprentissage est une fonction spécifiée par un paramètre $\theta \in \Theta$ appartenant à famille de fonctions paramétriques $\mathcal{H}$. Le risque peut être estimé par :

$$R(\theta) = \int Q((\mathbf{x}, y), \theta).dF(\mathbf{x}),\ \theta \in \Theta, \tag{5.2}$$

avec $Q((\mathbf{x}, y), \theta)$ est une fonction de coût (loss function).

$$Q((\mathbf{x}, y), \theta) = \ell((\mathbf{x}, y), h(\mathbf{x}, \theta)) \tag{5.3}$$

La fonctionnelle $R(\theta)$ est le **risque empirique** et correspond à la moyenne du coût, évaluation faite sur l'ensemble des exemples :

$$R_{emp}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell((\mathbf{x_i}, y_i), h(\mathbf{x_i}, \theta)) \tag{5.4}$$

Les fonctions de coût (loss functions)

La fonction de coût est une fonction $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ non négative et bornée, évaluée pour chaque objet, alors que le risque est évalué pour un ensemble. La Table 5.1 présente quelques exemples de fonction de coût usuelles.

	Fonction de Coût - $\ell(y_i, h(\mathbf{x_i}, \theta))$	Remarque
Norme L1	$\lvert h(\mathbf{x_i}, \theta) - y_i \rvert$	
Norme L2	$(h(\mathbf{x_i}, \theta) - y_i)^2$	
Coût *0-1*	$\mathbf{I}(h(\mathbf{x_i}, \theta) = y_i) = \begin{cases} 1 & \text{if } y_i = h(\mathbf{x_i}, \theta) \\ 0 & \text{if } y_i \neq h(\mathbf{x_i}, \theta) \end{cases}$	
Log Loss	$-log(p(y_i = h(\mathbf{x_i}, \theta) \mid \mathbf{x_i}))$	log vraisemblance

TAB. 5.1: Fonctions de coût usuelles. Pour la fonction de coût 0-1, le risque empirique correspond au taux moyen d'erreurs.

Sur-ajustement et régularisation

La minimisation du risque empirique évalué selon (5.4) présente l'inconvénient d'être appliqué sur les exemples, dont les valeurs associées sont connues, alors que ce qui nous intéresse est plutôt l'évaluation sur l'ensemble des objets passibles d'être rencontrés, c'est à dire, l'erreur de généralisation. Le risque empirique est donc optimiste. Minimiser le risque empirique sur les seuls exemples disponibles résulte généralement en sur-ajustement (overfitting), situation qui consiste à avoir une hypothèse parfaitement adaptée aux exemples traités mais pas aux objets devant être estimés.

Concrètement, le sur-ajustement résulte d'une hypothèse de dimension excessive[2].

[2]Les informaticiens regardent le problème de sur-ajustement comme un compromis entre l'erreur d'apprentissage et la complexité (dimension) du modèle, alors que les statisticiens "fréquentistes" le regardent comme un compromis entre la tendance (biais) et la variance [23, p. 147].

La première approche, la *Régularisation*, consiste à ajouter au risque empirique (5.4), un terme qui dépend de la complexité de l'hypothèse (modèle), qui devient [255] :

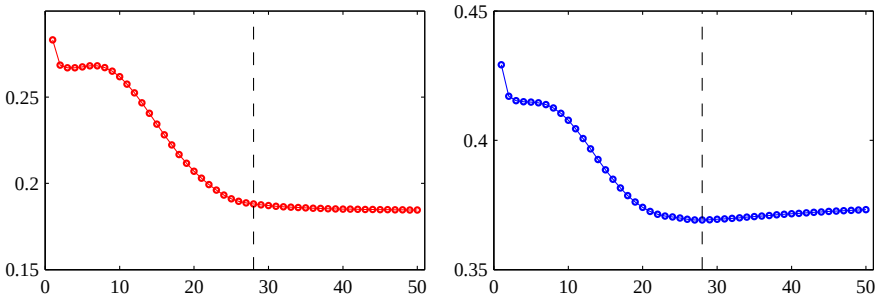
$$R^*(\theta) = R_{emp}(\theta) + \lambda(\delta) \cdot \Omega(\theta) \tag{5.5}$$

où $\Omega(\theta)$ est une fonction de la complexité (de la dimension) de l'hypothèse et $\lambda(\delta)$ est un facteur de pondération constant qui dépend du niveau de bruit dans les exemples. Un choix courant pour $\Omega(\theta)$ est la norme de θ, ce qui transforme (5.4) en :

$$R^*(\theta) = R_{emp}(\theta) + \frac{\lambda}{2} \cdot \|\theta\|^2 \tag{5.6}$$

D'autres approches de régularisation, toujours fondées sur la complexité du modèle, sont proposées par des critères tels BIC (Bayes Information Criteria), AIC (Akaike Information Criteria) et DIC (Deviance Information Criteria). Ces critères sont décrits en détail dans des textes tels [154], [211, chap 7], [126] ou [23] et brièvement dans l'Annexe A.

Un deuxième type d'approche, pratique, pour résoudre le problème de sur-ajustement est l'arrêt prématuré de l'apprentissage (voir Fig 5.1). Cette méthode consiste à évaluer simultanément l'erreur d'apprentissage sur les exemples et sur un ensemble de test et à arrêter l'apprentissage lorsque l'erreur sur l'ensemble de test atteint le minimum [23, p. 259]. D'autres approches pratiques, telles le *bootstrap* [96], qui consiste à ré-échantillonner avec remise les exemples d'apprentissage, sont aussi utilisées pour évaluer l'erreur de généralisation.



(a) Erreur d'apprentissage (b) Erreur de généralisation

FIG. 5.1: Arrêt prématuré de l'apprentissage - pour éviter le sur-ajustement, on vérifie, en même temps, l'erreur sur les exemples d'apprentissage et celui sur les cas de validation. L'apprentissage s'arrête lorsque l'erreur de généralisation atteint son point le plus bas. (figures tirées de [23])

5.3 Séparabilité

5.3.1 La notion de séparabilité

La notion de *classement parfait* dérive de la notion de *séparabilité* des ensembles, que l'on trouve en topologie. Nous allons utiliser une définition approximative, suffisante pour nos applications et pour comprendre l'idée dans le contexte de classement (pour une présentation plus rigoureuse, voir par exemple, [133, p. 78–85] [113, p. 145]).

Définition 5.1. Ensemble séparable - Un ensemble dénombrable $\mathcal{S}$ est dit séparable s'il existe au moins un partitionnement de $\mathcal{S}$ en N sous ensembles $\mathcal{S}_i$ non-vides et disjoints (par définition), tel que tout élément de $\mathcal{S}$ appartient à un et un seul des sous ensembles $\mathcal{S}_i$.

$$\begin{aligned} &\mathcal{S}_i \cap \mathcal{S}_j = \varnothing, \forall i, j \in \{1, \dots N\}, i \neq j \\ &\bigcup_{i=1}^{N} \mathcal{S}_i = \mathcal{S} \end{aligned} \tag{5.7}$$

Dans les problèmes de classement, le nombre de partitions est généralement fini et petit (2 dans le cas du classement de messages électroniques).

La séparabilité est une caractéristique essentielle dans un problème de classement. En effet, les données ne peuvent être classées sans erreur que quand ils peuvent être regroupés dans des ensembles séparables. Cette notion a été développée par Minsky et Papert dans le livre devenu célèbre *Perceptrons, An Introduction to Computational Geometry* [189], livre qui a démontré les limitations les limitations des *Perceptrons* en tant qu'algorithme de classement.

Les données peuvent être séparables, mais avoir des frontières complexes exigeant des classificateurs non linéaires [250]. Des frontières de séparation plus complexes exigent, en général, des classificateurs plus complexes.

Des ensembles discrets peuvent ne pas avoir la propriété de séparabilité soit parce que la représentation des éléments n'a pas la dimension suffisante - des informations concernant ces éléments ont été négligées, soit parce qu'il y a une raison non déterministe qui fait que l'attribution d'une classe à un élément n'est pas unique.

Mais il convient de souligner que dans un problème de classement la séparabilité, et donc la possibilité de classement sans erreur, est une caractéristique liée plutôt aux données qu'à l'algorithme de classement. Si, et seulement si, les ensembles de données sont séparables, il est toujours possible de trouver un algorithme capable d'effectuer un classement sans erreur.

5.3.2 Des sous-ensembles séparables par un classificateur

Définition 5.2. Sous-ensembles séparables par un classificateur - Soient $\mathcal{X}_i, i \in \{1, ..., N\}$ des sous-ensembles de vecteurs disjoints de $\mathcal{X}$. Ces sous ensembles sont séparables par une hypothèse $h : \mathcal{X} \mapsto \mathcal{Y}$, appartenant à une famille d'hypothèses $\mathcal{H}$, si et seulement si le classificateur évalue de façon identique les éléments appartenant à la même classe et différemment ceux appartenant à des classes différentes :

$$\forall \mathbf{x}_i \in \mathcal{X}_m, \mathbf{x}_j \in \mathcal{X}_n \qquad \begin{cases} m \neq n & \implies h(\mathbf{x}_i) \neq h(\mathbf{x}_j) \\ m = n & \implies h(\mathbf{x}_i) = h(\mathbf{x}_j) \end{cases} \tag{5.8}$$

Remarque 5.1. Représentation (dimension) insuffisante - Il peut arriver que certains objets ne puissent pas être classés, sans ambiguïté, dans une catégorie unique, mais que l'ajout d'une information permette de le faire. Par exemple, certains messages sont considérés comme spam par certains destinataires et comme messages légitimes par d'autres. Il suffirait d'ajouter un attribut pour désigner le destinataire, quand cela est possible, de façon à ce que les classes deviennent séparables.

5.3.3 Sous-ensembles linéairement séparables

Définition 5.3. Sous-ensembles linéairement séparables par un classificateur - Soit $\mathcal{X}$ un ensemble séparable et $\mathcal{X}_1$, $\mathcal{X}_2$ deux sous-ensembles disjoints de $\mathcal{X}$. Ces sous ensembles sont linéairement séparables par une hypothèse $h : \mathcal{X} \mapsto \mathcal{Y}$, s'ils sont séparables par h tel que défini en 5.2 et si $h(\mathbf{x})$ est une fonction linéaire.

Exemple 5.1. Hyperplan séparateur - Dans la présentation de la méthode des vecteurs de support, Vapnik [256, p.401] définit deux sous ensembles finis de vecteurs $\mathbf{x}$ d'un ensemble d'apprentissage :

$$\begin{aligned} \mathcal{X} &= \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}, \qquad \mathbf{x} \in \mathbb{R}^n,\ y \in \{-1, 1\} \\ \mathcal{X}_1 &= \{(\mathbf{x}, y) \in \mathcal{X} \mid y = -1\} \\ \mathcal{X}_2 &= \{(\mathbf{x}, y) \in \mathcal{X} \mid y = 1\} \end{aligned} \tag{5.9}$$

comme étant séparables par l'hyperplan

$$\langle \mathbf{x}, \phi \rangle = c \tag{5.10}$$

s'il existe un vecteur ϕ et une constante c tels que :

$$\begin{aligned} \langle \mathbf{x}, \phi \rangle &< c \text{ , si} \quad x \in \mathcal{X}_1 \\ \langle \mathbf{x}, \phi \rangle &> c \text{ , si} \quad x \in \mathcal{X}_2 \end{aligned} \tag{5.11}$$

Si bien que cet exemple a été employé dans le contexte d'étude des classificateurs à vecteurs de support, la définition n'est pas spécifique à ce type d'algorithme puisqu'elle dépend uniquement de la représentation des éléments des deux sous-ensembles.

Cet exemple permet distinguer deux entités :

- **Des sous ensembles de données linéairement séparables** - les classes pouvant être identifiées, sans erreur, par une hypothèse qui est une fonction linéaire des attributs ;
- **Des classificateurs linéaires** - des classificateurs pouvant être décrits par l'équation 5.10 (ou etre mis sous cette forme). Les classificateurs *Bayésien Naïf*, *Régression Logistique* et *Perceptron mono-couche* sont aussi des exemples de classificateurs linéaires.

La notion de séparabilité linéaire est intéressante puisqu'elle correspond, à la fois, à des classes plus facilement séparables et à des algorithmes de classement moins complexes.

5.4 Les modes d'apprentissage

Nous allons regrouper dans l'expression *"modes d'apprentissage"*, les différentes façons de faire interagir un algorithme d'apprentissage avec son environnement et de présenter des exemples à ces algorithmes. Ces modes d'apprentissage ne sont pas tous mutuellement exclusifs et peuvent, parfois, être combinés entre eux. Certains algorithmes de classement sont plus adaptés à certains modes d'apprentissage qu'à d'autres.

5.4.1 Apprentissage supervisé et non supervisé

Dans l'apprentissage *supervisé*, on utilise un ensemble d'exemples dont les valeurs associées sont connues à l'avance.

Dans le mode d'apprentissage *non supervisé*, les exemples sont présentés sans les valeurs associées. Ce mode est le plus souvent associé aux applications de regroupement (clustering), où l'objectif est plutôt de regrouper les objets selon leur ressemblance. On peut, par exemple, soumettre un ensemble de documents qui seront regroupés par thème et c'est à l'algorithme d'apprentissage de déceler les thèmes existants dans l'ensemble. Les applications qui nous concernent ne font pas usage de ce mode d'apprentissage et nous nous contentons de le définir.

L'apprentissage semi-supervisé est un mode intermédiaire où seule une partie des exemples a une classe associée.

5.4.2 Apprentissage hors ligne (batch) versus en ligne

Mitchell [190] définit *l'apprentissage en ligne* (online learning) comme celui fait à partir d'observations de résultats dans un environnement réel et l'*apprentissage hors ligne* (batch learning) celui résultant plutôt des observations obtenues à partir de simulations dans un modèle interne.

L'aspect temporel, mis en valeur par d'autres auteurs [239, p.241] [126, p. 355] [46] [245], nous semble particulièrement important à l'application objet de cette thèse.

Dans l'*apprentissage hors ligne* (ou *en batch*), les exemples sont entièrement traités dès le départ, avant toute opération de classement. L'hypothèse est entièrement construite, d'une seule fois, à partir des exemples disponibles. Cela suppose que les caractéristiques statistiques des objets restent stables dans le temps (il s'agit d'un processus stationnaire) ou alors que les dérives puissent être négligées.

Dans le contexte d'*apprentissage en ligne*, les exemples ne sont pas disponibles avant le départ des opérations d'estimation/classement. Les valeurs devant être associés à chaque exemple sont parfois disponibles après leur propre classement, sous la forme d'un retour d'information (boucle de rétroaction). A chaque nouvel exemple, une nouvelle hypothèse est générée, fonction de l'hypothèse précédente et de l'exemple en cours, ce qui suggère une application récursive.

$$h(i+1) = f(h(i), (\mathbf{x_i}, y_i))$$

Rien empêche qu'un processus stationnaire puisse être traité dans un contexte d'apprentissage en ligne mais, à cause du besoin d'adaptation aux changements, l'apprentissage en ligne d'un processus non-stationnaire devient absolument nécessaire.

On trouve, parfois des situations intermédiaires, appelées *"mini-batchs"*, où l'idée est que le processus évolue assez lentement pour qu'on puisse refaire l'apprentissage, périodiquement. En effet, il s'agit de considérer que localement, dans une fenêtre temporelle de taille adéquate, les changements ne sont pas trop importants. L'apprentissage est, donc, actualisé par pas de N exemples.

Hors ligne (batch)	**En ligne (stochastique)**
Conditions de convergence bien comprises.	Assez souvent, plus rapide que l'apprentissage hors ligne.
Certains algorithmes d'accélération de convergence ne sont applicables qu'aux algorithmes hors-ligne.	Meilleurs solutions, le plus souvent.
Analyse théorique de la dynamique des poids et de la vitesse de convergence moins complexe.	Utilisable pour suivre les changements dans un flot.
	Moins gourmand en ressources de calcul et de stockage.

TAB. 5.2: Comparaison des avantages des modes d'apprentissage *en ligne* et *hors ligne*

Dans un contexte d'apprentissage et de classement en ligne, les exemples étant présentés séquentiellement, l'information de l'étiquette associée peut ne pas être disponible immédiatement. Dans ce cas, il y a une dégradation de l'efficacité due au retard et, généralement, croissante avec ce paramètre.

Dans le Chapitre 8 nous examinons les raisons de les raisons de la dérive temporelle d'un flot de messages électroniques. Nous présentons dans le Chapitre 6 quatre algorithmes avec leurs particularités d'apprentissage en ligne. Nous présentons, très brièvement, dans l'Annexe C, un résumé des méthodes dites *approximations stochastiques*, que nous utiliserons dans nos expérimentations.

5.4.3 Apprentissage actif et apprentissage passif

Dans le mode d'*apprentissage passif*, un ensemble d'exemples est présenté à l'algorithme d'apprentissage, qui les utilise dans leur totalité. C'est l'environnement qui impose l'ensemble

des exemples à tenir compte.

L'*apprentissage actif* est un cas particulier d'*apprentissage supervisé* où seule une partie des exemples est utilisée. C'est à l'algorithme d'apprentissage de décider, avec une stratégie à définir, quels exemples choisir. C'est surtout cette capacité de questionner l'environnement qui le différencie de l'apprentissage passif.

Il a été démontré en pratique que juste une petite proportion d'exemples, avec les étiquettes associées, suffisent pour obtenir des niveaux d'efficacité comparables, et parfois meilleurs, que lorsque la totalité des exemples disponibles est utilisée [172] [251] [47].

Une stratégie naïve de sélection consiste à sous-échantillonner l'ensemble des exemples disponibles, par des tirages aléatoires indépendantes et identiquement distribués. Une stratégie plus efficace consiste à sélectionner les exemples selon un critère de gain. Par exemple, les plus informatifs,*c.à.d.* , les exemples qui viennent d'être classés mais dont le résultat sont les plus incertains [234].

Dans un contexte d'apprentissage hors-ligne, il s'agit de choisir dans un pool, les exemples les plus informatifs, tandis que dans un contexte d'apprentissage en ligne il s'agit juste de décider si un exemple doit être pris en compte ou pas.

Lewis et Gale [172] proposent de choisir, parmi les exemples disponibles, les b dont la prévision est la plus incertaine (uncertainty sampling). Si l'idée d'utilisation de l'incertitude est intéressante, l'inconvénient de cette stratégie est qu'elle examine, à chaque pas, l'ensemble des exemples disponibles pas encore utilisés, ce qui la rend plus adaptée à un contexte hors ligne que à contexte en ligne.

Cesa-Bianchi et al [47] proposent d'accepter un exemple avec une probabilité proportionnelle à $b/(b+|\hat{p}|)$, où b est un paramètre de réglage et $\hat{p}$ est la "distance" de l'exemple jusqu'à l'hyperplan défini par le classificateur (la marge).

Cette stratégie donne des résultats intéressants, mais l'auteur laisse le choix de la valeur de b comme un problème ouvert, avec quelques pistes. En effet, la notion d'utilisation d'une fonction monotone décroissante de la marge pour décider si l'on accepte un exemple correspond bien à une sélection par l'incertitude. Le problème plus général du choix optimal de cette fonction reste, à notre connaissance, un problème ouvert et il n'est même pas certain qu'un choix probabiliste tel celui proposé par Cesa-Bianchi donne, généralement, des résultats meilleurs qu'une valeur fixe de marge.

Seung et al [235] ont utilisé un comité d'experts pour sélectionner les exemples ayant reçu des prédictions discordantes (query by comitee). Ils ont rencontré, dans des cas synthétiques, un rapport qualitatif entre la quantité d'information et la marge d'incertitude sans toutefois généraliser à des contextes réels.

Tong et Koller [251] proposent une stratégie qui cherche à diminuer, à un moment donné, le nombre d'hypothèses plausibles (version space) - les hyperplans d'une SVM. Cette méthode aussi est peu adaptée à un mode d'apprentissage en ligne.

Un des objectifs souvent affichés de l'apprentissage actif est la minimisation du nombre d'exemples étiquetés. Il est connu que les algorithmes courants d'apprentissage artificiel donnent des résultats meilleurs lorsque les classes sont équilibrées en ce que concerne la taille de la population [99]. Ertekyn et all [99] ont démontré que l'apprentissage actif est capable de résoudre le problème d'asymétrie des classes justement parce les exemples sont choisis de façon équilibré, ne prenant que les exemples pertinents.

Burr Settles [234] maintien une référence bibliographique intéressante concernant l'apprentissage actif.

Enfin, vue la difficulté d'obtenir des exemples de messages électroniques, avec les classes associées, ce mode d'apprentissage semble particulièrement intéressant pour l'application de filtrage de spam.

5.5 Conclusions

Dans ce chapitre nous avons résumé brièvement la problématique d'apprentissage artificiel. En particulier, nous avons montré le concept de séparabilité, essentiel pour qu'un classement sans erreur soit possible. Nous avons aussi présenté la problématique de sur ajustement (overfitting). Dans un contexte d'apprentissage en mode batch, ce problème est résolu par régularisation ou arrêt précoce d'apprentissage, méthodes qui s'appliquent moins bien lorsque l'apprentissage se fait en ligne, puisque le flot d'objets n'est pas statique.

Dans un tutoriel sur le filtrage de spam, Joshua Goodman [114, p. 40] signale quelques erreurs commises usuellement dans l'évaluation de filtres anti-spam et en particulier des méthodes, telles la validation croisée, utilisées dans l'évaluations de processus hors ligne. Gordon Gormack et Thomas Lynam [69] ont comparé expérimentalement les résultats de plusieurs filtres distribués sous licence libre ainsi que certaines publications de résultats de recherche et ont remarqué des nombreuses incohérences dues, en particulier, à l'utilisation d'ensembles de messages inadaptés ou alors à des simulations d'applications hors ligne alors que le filtrage de spam est un processus en ligne. Avec Andrej Bratko, Gordon Cormack [60] a comparé les résultats de quelques méthodes de filtrage de spam dans un environnement en ligne avec celui hors ligne et a obtenu, en général, des résultats optimistes dans les simulations hors ligne.

Il est maintenant acquis que le filtrage de spam doit être étudié comme étant un processus en ligne, puisqu'il s'agit d'un processus évolutif vraisemblablement non stationnaire.

Le mode d'apprentissage qui semble intéressant dans la problématique de classement de messages électroniques est l'apprentissage actif, à cause de la rareté d'échantillons étiquetés. En effet, ce mode d'apprentissage cherche à n'utiliser que les échantillons pouvant améliorer effectivement la qualité de classement.

CHAPITRE 6

Les Algorithmes de Classement

True ignorance is not the absence of knowledge, but the refusal to acquire it.

Karl Raimund Popper

6.1 Introduction

Dans ce chapitre nous présentons quatre algorithmes de classification utilisés/utilisables pour le classement de spam. Le premier, le classificateur *Bayésien Naïf* n'est pas le plus performant, mais c'est le plus simple à mettre en oeuvre et le plus diffusé, en particulier dans les logiciels libre de filtrage de *spam*. Les autres algorithmes - *régression logistique* et *SVM* (machines à vecteur de support) sont ceux qui se sont montrés les plus performants lors des expérimentations menées dans le dernier TREC Spam Track (2007). Le *Perceptron* est le plus simple et le plus ancien de tous - il a servi d'inspiration pour la construction du classificateur qui sera utilisé dans ces travaux.

6.2 Le classificateur Bayésien

Le classificateur Bayésien trouve ses racines dans le Théorème de Bayes. Le problème général du classement peut-être posé comme le choix de la *meilleure hypothèse* associée à un objet, après observation d'un ensemble d'exemples d'apprentissage. Une façon de définir la *meilleure hypothèse* est de considérer celle qui est la *plus probable*, c'est-à-dire celle dont la probabilité d'erreur est la plus faible.

Considérons $P(c)$, $c \in \mathcal{Y} = \{spam, nospam\}$ la probabilité associée à une hypothèse de classement, avant que l'objet à classer (un message) ne soit observé. Il s'agit de la *Probabilité à Priori* de l'hypothèse. Après avoir observé un message reçu M, la probabilité à posteriori peut être évaluée selon la règle de Bayes pour chaque classe :

$$P(Y = c \mid M = m) = \frac{P(M = m \mid Y = c)\, P(Y = c)}{P(M = m)},\ c \in \mathcal{Y} \tag{6.1}$$

La comparaison des probabilités à posteriori des classes candidates définit la règle de décision optimale comme étant le choix de la classe qui maximise cette probabilité :

$$\begin{aligned}\hat{y} &= \underset{c \in \mathcal{Y}}{\arg\max} \ \frac{P(M \mid c)\ P(c)}{P(M)} \\ &= \underset{c \in \mathcal{Y}}{\arg\max} \ (P(M \mid c)\ P(c))\end{aligned} \tag{6.2}$$

Le dénominateur peut être négligé puisqu'il s'agit d'une valeur qui reste constante quand évaluée pour chacune des classes.

L'estimation des probabilités à priori des classes $P(c)$ ne pose pas des difficultés : quelques centaines suffisent pour obtenir une précision suffisante. L'estimation des probabilités à postériori $P(M \mid c)$ est un problème intraitable à cause du nombre de paramètres devant être estimés et d'exemples nécessaires [190, p. 180]. La solution alternative est de considérer *naïvement* (d'où le nom *Bayésien Naïf*) que tous les attributs sont indépendants les uns des autres. Cette hypothèse fait que les probabilités associées à chaque terme peuvent être estimées individuellement. Malgré l'apparente faiblesse de cette hypothèse elle donne des très bons résultats en pratique qui s'expliquent par le fait que la tâche de classement utilisant des attributs binaires dépend plus du signe d'une fonction d'estimation (hypothèse) que de l'exactitude [90] [105] [185].

L'évaluation de la probabilité du message $P(M \mid c)$ dépend du modèle événementiel, objet de la prochaine section.

6.2.1 Modèles événementiels

Dans les classificateurs bayésiens naïfs, le modèle du processus de génération des objets définit comment évaluer la probabilité conditionnelle sachant la classe, à partir des probabilités individuelles des termes présents dans le message.

Dans ces modèles on considère que l'objet est généré à partir d'un mélange de n distributions de probabilité (n est le nombre de classes). Le processus consiste à, d'abord, choisir une classe au hasard avec probabilité $P(c)$ (la probabilité à priori de la classe) et ensuite générer un message selon la distribution des termes spécifique à la classe choisie. C'est pour cette raison que l'on dit que le classificateur *Bayésien Naïf* est un classificateur *génératif*.

Les modèles les plus courants de génération de documents textuels sont Multivariate Bernouilli et Multinomial [185] [224] [188]. D'autres modèles permettent le traitement d'attributs continus tels Flexible Bayes et Multivariate Gaussian [140] mais leur complexité et efficacité relative ne semblent pas justifier leur utilisation dans les filtres anti-spam.

Dans la présentation de ces modèles nous utilisons la convention :

$$\mathcal{S} = \{(\mathbf{m_1}, y_1), ..., (\mathbf{m_N}, y_N)\},\ (\mathbf{m_i}, y_i) \in \mathcal{M} \times \mathcal{Y} \tag{6.3}$$

est un échantillon de N exemples de $\mathcal{M}$ utilisés pour l'apprentissage. $\mathcal{V} = \{t_1, t_2, \ldots, t_{|\mathcal{V}|}\}$ est le vocabulaire, l'ensemble de tous les termes que l'on peut rencontrer dans tous les objets. $\mathbf{m} = (m_1, m_2, \ldots, m_\ell)$ est un message avec ℓ termes. $\mathbb{I}_c(c_j, \mathbf{m_i})$ est une fonction indicatrice de la classe d'appartenance de l'exemple $\mathbf{m_i}$.

$$\mathbb{I}_c(\mathbf{m_i}, \mathbf{c_j}) = \begin{cases} 1 & \text{si } y_i = c_j \\ 0 & \text{si } y_i \neq c_j \end{cases} \tag{6.4}$$

Les fonctions indicatrices $\mathbb{I}_p(t, \mathbf{m})$ et $\mathbb{I}_n(t, \mathbf{m})$ indiquent, respectivement, la présence et le nombre d'occurrences du terme t dans l'objet $\mathbf{m}$.

Multivariate Bernouilli

Dans ce modèle l'objet résulte de plusieurs tirages au sort : un pour chaque terme du vocabulaire, pour décider si le terme en question est présent dans l'objet. Chaque tirage suit une loi de Bernouilli de probabilité $P(t_j|c)$. La probabilité, dans chaque classe, du message généré de cette façon est estimée par :

$$P(\mathbf{M} \mid c) = \prod_{t \in \mathcal{V}} \left(P(t \mid c)^{\mathbb{I}_p(t,\mathbf{M})} \, (1 - P(t \mid c))^{(1-\mathbb{I}_p(t,\mathbf{M}))} \right) \tag{6.5}$$

La distribution des termes dans les classes $P(t \mid c)$ est généralement évaluée à l'aide d'une approximation de Laplace ou de Lidstone[1]. Avec l'approximation de Laplace, on a :

$$\hat{P}(t \mid c) = \frac{1 + \sum_{\mathbf{m} \in \mathcal{S}} \mathbb{I}_p(t, \mathbf{m}) \, \mathbb{I}_c(\mathbf{m}, c)}{2 + \sum_{\mathbf{m} \in \mathcal{S}} \mathbb{I}_c(\mathbf{m}, c)}, \qquad t \in \mathcal{V},\, c \in \mathcal{Y} \tag{6.6}$$

A noter que l'évaluation de $P(\mathbf{M} \mid c)$ tient compte de tous les termes du vocabulaire, ce qui peut être couteux pour un vocabulaire de grande taille.

Multinomial

Dans ce modèle, on considère que le message est généré par un nombre limité ℓ, la longueur du message, de tirages aléatoires, avec remise, des termes d'un vocabulaire. Chaque terme peut apparaitre plus d'une fois. La probabilité du message dans chaque classe est estimée par :

$$P(\mathbf{M} \mid c) = P(|\mathbf{M}| = \ell \mid c) \; \ell! \prod_{t \in \mathcal{V}|} \frac{P(t \mid c)^{\mathbb{I}_n(t,\mathbf{m})}}{\mathbb{I}_n(t, \mathbf{m})!} \tag{6.7}$$

Comme dans le cas précédent, on utilise une approximation de Laplace pour estimer les probabilités de chaque terme dans sa classe.

$$\hat{P}(t \mid c) = \frac{1 + \sum_{\mathbf{m} \in \mathcal{S}} \mathbb{I}_n(t, \mathbf{m}) \, \mathbb{I}_c(\mathbf{m}, c)}{|\mathcal{V}| + \sum_{\mathbf{s} \in \mathcal{V}} \sum_{\mathbf{m} \in \mathcal{S}} \mathbb{I}_n(s, \mathbf{m}) \, \mathbb{I}_c(\mathbf{m}, c)}, \qquad t \in \mathcal{V},\, c \in \mathcal{Y} \tag{6.8}$$

Au contraire do modèle multivariate, dans le modèle multinomial seuls les termes présents dans le message sont pris en compte dans le produit qui n'a donc que ℓ termes, au plus.

Multinomial avec attributs booléens

C'est un cas identique au cas multinomial mais on ne considère que la présence ou absence des termes.

$$P(\mathbf{M} \mid c) = P(|\mathbf{M}| = \ell \mid c) \; \ell! \prod_{t \in \mathcal{V}} P(t \mid c)^{\mathbb{I}_p(t,\mathbf{m})} \tag{6.9}$$

La probabilité de chaque terme dans sa classe est estimée par :

$$\hat{P}(t \mid c) = \frac{1 + \sum_{\mathbf{m} \in \mathcal{S}} \mathbb{I}_p(t, \mathbf{m}) \mathbb{I}_c(\mathbf{m}, c)}{|\mathcal{V}| + \sum_{\mathbf{s} \in \mathcal{V}} \sum_{\mathbf{m} \in \mathcal{S}} \mathbb{I}_p(s, \mathbf{m}) \, \mathbb{I}_c(\mathbf{m}, c)}, \qquad t \in \mathcal{V},\, c \in \mathcal{Y} \tag{6.10}$$

Discussion

McCallum et Nigam [185] ont démontré empiriquement que, dans des applications de classement textuel, le modèle multivariate ne donne des meilleurs résultats que pour vocabulaires de très petite taille (quelques centaines de termes). Schneider [224], Metsis et al [188] ont démontré empiriquement que, dans le cas du classement de spams, le modèle multinomial est plus efficace. Ceci n'est pas étonnant puisque certains mots semblant être des forts indicateurs de spam, *e.g. viagra*, peuvent indiquer plutôt une discussion entre médecins (donc un *ham*) quand ils apparaissent plusieurs fois dans le même message. *c.à.d.* le nombre d'apparitions d'un terme n'est pas toujours un indicateur pertinent.

[1] Voir Annexe A

6.2.2 Apprentissage et implémentation

L'apprentissage d'un classificateur bayésien naïf se fait tout simplement par le comptage des termes trouvés dans chacune des classes dans un corpus d'apprentissage. L'estimation des probabilités à posteriori de chaque terme sachant la classe peut se faire au moment de son utilisation.

Dans les filtres anti-spam, la pratique courante est d'effectuer la sélection d'attributs en deux étapes. Lors de l'apprentissage, seuls sont retenus les attributs dont le nombre d'apparitions dans le corpus dépasse un seuil dépendant de la taille du corpus. Lors des opérations de classement, on sélectionne les N termes les plus significatifs, selon un critère tel le gain d'information ou équivalent. Ce nombre varie selon l'implémentation du classificateur, mais se situe entre quelques dizaines à quelques centaines.

6.2.3 Classificateur Linéaire

Les classificateurs linéaires sont intéressants puisque leur analyse est souvent plus simple. Certains modèles événementiels génèrent des classificateurs qui peuvent être mis sous forme linéaire. C'est le cas du modèle multinomial avec des attributs binaires.

Pour une catégorisation en deux classes $\mathcal{Y} = \{c_1, c_2\}$, la fonction logarithme étant une fonction monotone croissante, la règle de décision (6.2) peut être exprimée sous la forme :

$$\hat{y} = \begin{cases} c_1 & \text{si } h(\mathbf{M}) \geq 0 \\ c_2 & \text{si } h(\mathbf{M}) < 0 \end{cases} \qquad h(\mathbf{M}) = \log \frac{P(M \mid c_1)\ P(c_1)}{P(M \mid c_2)\ P(c_2)} \tag{6.11}$$

Si l'on représente le message sous la forme d'un vecteur dont la dimension est celle du vocabulaire $\mathbf{X} = (x_1, x_1, \ldots, x_{|\mathcal{V}|})$, chaque composante indiquant la présence ou absence du terme correspondant dans le message : $x_i = \mathbb{I}_p(t_i, \mathbf{M}), i = 1, \ldots, |\mathcal{V}|$.

Dans le cas du modèle événementiel multinomiale avec des attributs booléens le remplacement de (6.9) dans (6.11) permet d'obtenir $h(\mathbf{X})$ comme une fonction linéaire des attributs de $\mathbf{X}$:

$$h(\mathbf{X}) = \log \frac{P(|\mathbf{X}| = \ell \mid c_1)}{P(|\mathbf{X}| = \ell \mid c_2)} + \log \frac{P(c_1)}{P(c_2)} + \sum_{i=1}^{|\mathcal{V}|} x_i \log \frac{P(t_i \mid c_1)}{P(t_i \mid c_2)} \tag{6.12}$$

Le premier terme, dépendant de la longueur, est généralement négligé soit parce que la longueur des documents est normalisée, soit parce que seule une quantité fixe d'attributs est retenue.

6.2.4 Les classificateurs bayésien naïfs et les logiciels libres

Dans la communauté des logiciels libres, quasiment tous les filtres fondés sur une apprentissage statistique sont désignés par la dénomination *"Filtres bayésiens"*, alors que la plupart ne le sont pas. Ces filtres se sont inspirés d'une idée proposé par Paul Graham dans un article publié dans son blog [2], se basant sur la règle de Bayes.

La première différence par rapport aux classificateurs bayésiens que l'on trouve dans les publications scientifiques est qu'aucun de ces filtres n'intègre la probabilité à priori des classes et la considèrent comme ayant une valeur fixe, alors que l'on constate que cette probabilité varie considérablement selon l'heure de la journée.

Il y a eu des nombreuses discussions sur la façon de combiner les probabilités des termes pour obtenir la probabilité conditionnelle du message sachant la classe. Certains filtres se sont inspirés d'un article publié dans le blog de Gary Robinson[3] indiquant que les probabilités des termes doivent être combinées sous la forme d'un produit qui aurait une distribution de χ^2. Zdziarski [273, p. 215] décrit *Fifth Order Markovian Discrimination* qui, au contraire de ce que dit l'auteur, ne serait pas une méthode de classement mais plutôt la représentation d'un message

[2] A Plan for Spam - `http://www.paulgraham.com/spam.html`
[3] A Statistical Approach to the Spam Problem - `http://www.linuxjournal.com/article/6467`

par les combinaisons possibles de deux mots pris dans une fenêtre glissante de taille 5 mots, et que tiendrait compte de la dépendance des mots, telle une chaîne de Markov. On retrouve cette représentation dans le filtre *CRM114*[4], sous la dénomination *OSBH* (Orthogonal Sparse Binary Hash).

Malgré l'empirisme de la majorité de ces filtres, ils ont permis une meilleure connaissance de l'environnement de filtrage de spam, grâce à une très large diffusion et à des expérimentations avec des données réels.

6.2.5 Discussion et Conclusions

Le classificateur bayésien naïf, par la simplicité d'implémentation, l'efficacité et le faible coût de traitement informatique, est le plus largement diffusé.

L'apprentissage de ce filtre consiste, comme nous l'avons vu, à comptabiliser, pour chaque terme du vocabulaire, le nombre de messages dans chaque classe où ce terme apparait. Ceci

L'expérience pratique, en fonctionnement réel, que nous avons avec ce type de filtre montre que l'efficacité se dégrade avec le temps. Plusieurs peuvent être les causes, mais la cause la plus probable est le surajustement (overfit) résultant de l'ajout, à la longue, de nouveaux exemples mal classés, sans suppression des exemples les plus anciens. En effet, comme nous verrons dans le Chapitre 8, l'apprentissage en ligne exige que l'on oublie les messages les plus anciens au fur et à mesure de l'avancement du temps et de l'ajout de nouveaux exemples. Le besoin de maintenir la comptabilité des termes nécessite le stockage des messages encore valables et constitue un inconvénient pour l'apprentissage en ligne.

6.3 Le *Perceptron*

Il s'agit d'un algorithme d'apprentissage permettant de résoudre efficacement des problèmes linéaires de classement et proposé initialement par Rosenblatt en 1957 et publié en 1962 [217] [216].

Le *Perceptron* a beaucoup intéressé la communauté d'intelligence artificielle jusqu'à la publication du livre de Minsky et Papert [189] qui démontrait ses avantages et surtout ses inconvénients : l'impossibilité de séparer des classes définies par des fonctions non linéaires, *e.g.* un *ou-exclusif*. Ceci est du au fait que l'interprétation de l'algorithme de classement du Perceptron est géométrique : l'équation $\langle \mathbf{w} \cdot \mathbf{x} \rangle = 0$ défini l'hyperplan séparant les deux classes. L'intérêt par le *Perceptron* a repris dans les années 80.

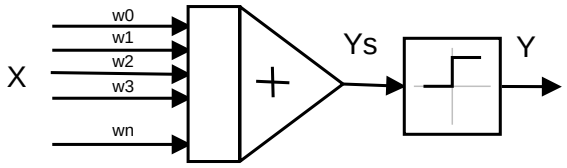


FIG. 6.1: Le Perceptron

En mode classement, le fonctionnement du *Perceptron* est décrit par l'équation 6.13 [23] [155], où $\mathbf{x}$ est un vecteur représentant un objet à classer et $\mathbf{w}$ est le vecteur des paramètres du *Perceptron*. Le signe du résultat indique la classe à attribuer à l'objet (l'appartenance à chaque classe est représentée par les valeurs -1 et 1).

$$\hat{y} = sign\left(\langle \mathbf{w}, \mathbf{x} \rangle\right) = \begin{cases} +1 & \text{si } \langle \mathbf{w}, \mathbf{x} \rangle \geq 0 \\ -1 & \text{si } \langle \mathbf{w}, \mathbf{x} \rangle < 0 \end{cases} \tag{6.13}$$

[4]CRM114 - `http://crm114.sourceforge.net`

L'apprentissage consiste à trouver le paramètre $\mathbf{w}$ annulant l'erreur globale. Plusieurs variantes d'algorithme d'apprentissage ont été proposées.

Dans la version ci-dessous (Algorithme 6.1), la plus élémentaire, l'ensemble d'exemples est examiné entièrement, jusqu'à ce que tous les exemples soient classés correctement. A chaque exemple examiné, si le classement est erroné, le vecteur de poids $\mathbf{w}$ est actualisé. Cet algorithme converge en un nombre fini d'itérations, à condition que les classes soient linéairement séparables, mais la convergence peut être très lente.

Algorithme 6.1 Apprentissage hors ligne du Perceptron

```
w ← 0
repeat
  k ← 0
  for all x_i ∈ S do
    if ŷ_i ≠ y_i then
      w ← w + η (y_i − ŷ_i) x_i
      k ← k + 1
    end if
  end for
until k = 0 (plus d'erreurs)
```

Un autre algorithme similaire, *Adaline* (Adaptive Linear Element), a été développé par l'équipe de Widrow en même temps que le Perceptron, mais avec un approche d'apprentissage différente, connue sous la dénomination "*delta rule*" de Widrow et Hoff [263].

6.4 Régression Logistique

Comme pour le classificateur Bayésien, l'algorithme de Régression logistique a une interprétation probabiliste puisque sa décision de classement est fondée sur la probabilité à posteriori de la classe.

Ce modèle considère que, sous des hypothèses assez générales, le logarithme de la vraisemblance à posteriori peut être écrit comme une fonction linéaire du vecteur d'attributs de l'objet à classer. Ainsi, la probabilité à postériori peut être décrite par une fonction sigmoïde agissant sur le vecteur d'attributs [23, p. 205] :

$$P(spam \mid \mathbf{m}) = \frac{1}{1 + e^{-\langle \mathbf{w}, \mathbf{m} \rangle}} \tag{6.14}$$

ce qui permet d'évaluer directement la probabilité à postériori de la classe, sans simulation du processus de génération de l'objet, comme c'est le cas pour l'algorithme Bayésien. L'apprentissage de cet algorithme consiste alors à déterminer le vecteur $\mathbf{w}$.

L'apprentissage d'un classificateur à régression logistique fait appel, en général, à des méthodes itératives tels GIS (Generalized Iterative Scaling) [84] proposé par Darroch et Ratcliff en 1972 ou des algorithmes plus récents tels IRLS (Iterative Reweighted Least Squares) [23, p. 207].

Certains auteurs [255, p. 156] [155, p. 267] suggèrent que l'apprentissage de certains algorithmes, comme par exemple un *Perceptron* modifié utilisant la fonction sigmoide comme fonction de lien (link function), convergent vers la même solution que le classificateur à régression logistique. Néanmoins, ces algorithmes ont des différences conceptuelles importantes. L'interprétation du *Perceptron* est géométrique : un hyperplan de séparation entre deux classes linéairement séparables, tandis que l'interprétation de la Régression Logistique est probabiliste. Ces deux algorithmes ne sont équivalents que dans des contextes particuliers.

Ce classificateur, avec apprentissage supervisée, a été proposé par Goodman [115] et utilisé par Cormack [58] [57] dans TREC 07.

6.5 Machines à Vecteur de Support - (SVM)

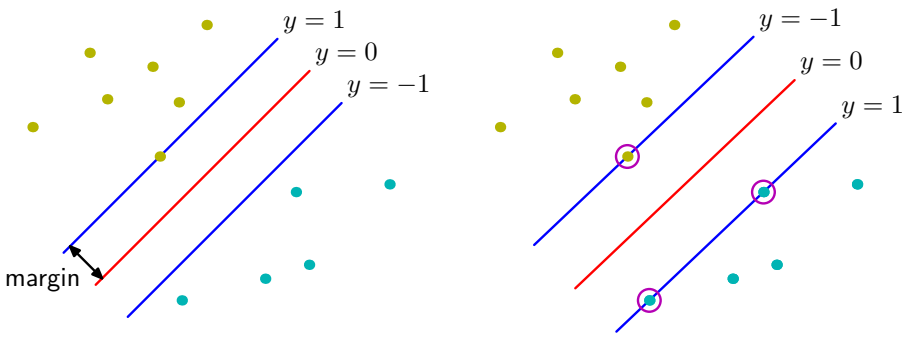


FIG. 6.2: Dans un SVM, ou Machine à Vecteurs de Support, la marge, ou distance entre le hyperplan séparateur et les exemples les plus proches, est maximale.

Il s'agit d'un classificateur avec une interprétation géométrique (voir Figure 6.2 et s'appuie sur la notion de *marge maximale*. La *marge* est la distance qui sépare la frontière de séparation (un hyperplan) des exemples les plus proches. Les vecteurs définissant la distance entre entre la frontière et les exemples les plus proches sont les *Vecteurs de Support*. L'apprentissage consiste à trouver l'hyperplan assurant une marge maximale, à partir de l'ensemble d'exemples, dont la solution est un problème d'optimisation quadratique. On peut utiliser ce type de classificateur pour résoudre des problèmes non linéaires par projection dans un espace de dimension supérieure. Pour plus de détails sur ce type de classificateur voir, par exemple, [256] [77] [225] [239] ou [136] pour l'application aux problèmes de classement textuel.

Les SVM ont été proposées pour le classement de messages électroniques dès 1999 par Drucker et all [92]. Sculley a proposé ROSVM [229] [230] [226] pour le classement de *spam* et apprentissage en ligne (en mini-batches). Il s'agit d'une version allégée, où le nombre d'itérations de l'algorithme d'optimisation est limité ainsi que le nombre d'exemples dans une fenêtre temporelle.

6.6 Notes bibliographiques

La bibliographie concernant les algorithmes d'apprentissage artificielle est assez vaste. Des livres intéressants sont, par exemple, Mitchell [190], [209], Hastie, Tibshirani et Friedman [126], Bishop [23], Kononenko et Kukar [155]. La référence la plus intéressante couvrant l'application de ces algorithmes au problème spécifique de classement de messages électroniques est la monographie de Gordon Cormack [59].

Troisième partie

Mutualisation du classement de messages électroniques

CHAPITRE 7

L'utilisation mutualisée d'un filtre anti-spam

La pensée n'est qu'un éclair au milieu de la nuit. Mais c'est cet éclair qui est tout.

Henri Poincaré, La valeur de la science

7.1 Introduction

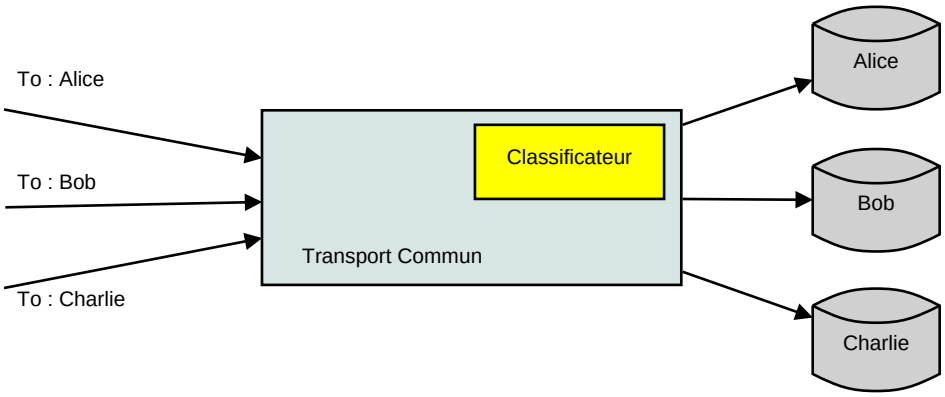


FIG. 7.1: Classement mutualisé des messages adressés à plusieurs destinataires, utilisant des ressources communes.

Lorsque le flot de messages de plusieurs destinataires traverse une même série de noeuds de traitement, on est tenté d'utiliser aussi des ressources et paramètres communs pour effectuer le classement des messages et l'élimination des spams. Ceci est schématisé dans la Figure 7.1.

Il semble exister un consensus sur le fait que, en pratique, l'utilisation d'un classificateur statistique pour filtrer collectivement les messages d'un groupe d'individus est une situation problématique [149] [249] [261] [233]. Ce consensus résulte de l'idée que les flots de messages de destinataires différentes ont des caractéristiques différentes, ce qui est surement vrai, chaque classificateur étant adapté à un flot spécifique n'est pas adapté au flot combiné. Ces travaux posent des questions pertinentes mais ils considèrent un problème binaire : utilisation individuelle ou collective. Les solutions proposées sont expérimentées dans le contexte particulier, sous-entendu,

de l'étude. Il n'y a pas, ou pas assez, à notre connaissance, de travaux publiés pour confirmer ou infirmer ce consensus, se basant sur les caractéristiques réelles des flots.

En fait, on peut identifier, par leurs caractéristiques, au moins trois grandes catégories de collectivités partageant un même service de messagerie. Ces caractéristiques font que l'on peut penser qu'avoir une approche différente selon la catégorie de collectivité nous permettrait d'atteindre plus facilement un filtrage plus performant, en particulier dans la catégorie qui nous intéresse : typiquement une université.

Dans beaucoup des travaux publiés, on se concentre le plus souvent sur l'évaluation des algorithmes de classement proposés, sans démontrer l'adéquation entre les corpus de messages de validation et le contexte d'utilisation. Il est souvent assumé, explicitement ou pas, dans les travaux utilisant des corpus publiques de messages, que le résultat de classement ne dépend pas de son destinataire [66] [56].

L'objectif de ce chapitre est d'identifier les caractéristiques de ces catégories et d'imaginer les types de solutions applicables, en particulier pour la catégorie qui nous intéresse.

7.2 La taxonomie des communautés

Il est tentant d'essayer de résoudre globalement la problématique d'utilisation d'un filtre antispam, quelque soit le type de communauté. Néanmoins, nous identifions trois grandes catégories de communautés, avec des caractéristiques différentes :

- **Les Hébergeurs de Boïtes aux Lettres -** (ou *ESP* - E-mail Service Provider) Ce sont des services gratuits ou payants (tels *Yahoo*, *Gmail*, *Hotmail*, *FastMail*, ...). La caractéristique première de ces services est l'absence, à priori, de liens entre les abonnés ou alors des liens très faibles, parfois contractuels, entre les abonnés et le prestataire. Dans ces conditions, aucune inférence peut être faite sur la constitution des messages légitimes envoyés ou reçus par les abonnés.
- **Les entreprises -** C'est, parmi ces trois types, la structure la plus rigide : les activités des employés, ainsi que le contenu habituel des messages électroniques, sont bien définis et connus, à tel point qu'il est possible de regrouper les employés en un nombre limité et faible de profils assez précis. Les services administratifs ont, en plus, le pouvoir de décider de la stratégie de filtrage, parfois triviale, imposée aux employés.
- **Les communautés d'enseignement et recherche -** Il s'agit d'une communauté dont la diversité est intermédiaire entre les entreprises et les hébergeurs de messagerie. La connaissance que l'on peut avoir de la constitution des messages échangés est assez vague, mais étonnante. Vue la liberté intrinsèque des chercheurs et enseignants, les messages, à première, vue étonnants ne sont pas forcément illégitimes. Aussi, le réseau de correspondants est beaucoup plus large, mais avec un nombre souvent plus important d'interlocuteurs communs, avec une participation souvent plus importante dans des listes de discussion et adresses de messagerie collectifs.

Le tableau (7.1) présente des caractéristiques de chacune de ces communautés.

7.3 L'interaction avec les destinataires

L'apprentissage d'un classificateur dépend entièrement du jugement des messages par le destinataire, donc de l'existence d'un retour d'information sur le classement des messages déjà reçus. Cela pose des problèmes les plus divers, y compris dans un environnement de filtrage mono-utilisateur, où le filtre est installé sur son poste de travail et sous son entière responsabilité[1]. Des exemples de points importants sont :

[1] Par exemple, avec un logiciel du type *Thunderbird* où l'utilisateur a un petit bouton à cliquer lui permettant de signaler si un message est légitime ou pas.

	Hébergeur de Boïtes aux Lettres	**Entreprise**	**Communautés E/R**
Utilisation	Plutôt personnelle.	Exclusivement professionnelle.	Professionnelle et personnelle.
Diffusion des adresses à l'extérieur	Faible, à l'initiative des abonnés : dans des forums ou des blogs, par exemple.	Très faible, souvent limitée aux services commerciaux, sous la forme d'adresses génériques.	Large : sur pages web personnelles, publications, conférences, forums, ...
Pouvoir de censure de l'administrateur de la messagerie	Aucun qui ne puisse pas être justifié par des raisons de sécurité.	Fort.	Aucun et les raisons de sécurité ne justifient pas.
Connaissance de la constitution des messages et du réseau de correspondants	Faible.	Forte.	Moyenne.
Partage de caractéristiques entre destinataires	Aucun.	Fort.	Moyen.
Ressemblance des boîte aux lettres des utilisateurs	Aucune.	Forte.	Moyenne.
Topologie du réseau de correspondants et des échanges	Correspondants plutôt extérieurs.	Communication plutôt interne.	Correspondants extérieurs et intérieurs.
Diversité Linguistique	Souvent forte.	Souvent faible.	Souvent forte.
Adresses alternatives ou de regroupement	Un individu = une adresse.	Une adresse personnelle plus une éventuelle de contact.	Plusieurs.
Définition de *spam*	Générique, acceptée par l'utilisateur.	Stricte, imposée par l'entreprise.	Variable selon l'utilisateur.

TAB. 7.1: Les caractéristiques systèmes de messagerie selon le type de communauté

Ergonomie et non-intrusion - il faut que l'utilisateur puisse, en un ou deux "clics de souris", démarrer le processus de retour d'information sans qu'il ne soit dérangé dans son activité en cours. Il ne faut pas que cela soit une contrainte.

Exhaustivité - les utilisateurs ne signalent pas systématiquement les erreurs ou l'exactitude de classement et quand ils le font c'est, le plus souvent, pour une seule des classes. Le signalement systématique de bon classement de tous les messages n'est pas envisageable. Donc, l'apprentissage se fait avec un ensemble de messages qui ne correspond pas à un échantillonnage *i.i.d.* du flot de messages.

Confidentialité - en général, pour des raisons de protection de la vie privée, les utilisateurs n'acceptent pas de transmettre leurs messages légitimes à un administrateur de messagerie, alors qu'il s'agit des messages qu'ils ne souhaitent surtout pas voir classés en erreur. Les utilisateurs acceptent plus facilement de fournir des retours d'information sur les *spams* que sur les *hams*.

Diversité des critères d'appréciation - La définition de ce que c'est un *"spam"* est déjà assez floue et varie selon le point de vue légal, ou technique mais varie aussi d'individu à individu, aussi bien du point de vue définition que du seuil de tolérance au *spam*.

Retard - Il y a toujours un retard entre le moment où le message passe par le filtre et le moment où le destinataire lit le message et retourne une information de classement. Selon la *"distance"* entre le filtre et le destinataire, ce retard peut varier entre quelques minutes et quelques jours. Ce retard peut aussi varier selon l'heure de la journée et le jour de la semaine. Ce délai peut-être même exceptionnellement très long : *e.g.* pendant les vacances.

7.4 La taxonomie des solutions de filtrage mutualisé

Le type de solution probablement le plus largement diffusé et utilisé pour le filtrage de spam dans une communauté, ce sont les listes noires ou de réputation. Il s'agit de listes d'adresses IP considérées suspectes. Cette méthode ne relève pas des techniques d'apprentissage artificiel et se caractérise parle fait que le filtrage est fondé uniquement sur le mode de distribution des messages et ne tient pas compte ni du contenu, ni du destinataire. Il ne s'agit pas d'une opération de *classement* en deux catégories mais juste d'*identification* des messages distribués par des intermédiaires jugés suspects. Ce sont des solutions "prête à l'emploi" avec un fonctionnement indépendant du type de communauté. Néanmoins, le taux de détection de spams de ce type de solution rarement dépasse 80 % sur un flot de messages réel [119]. Le fonctionnement de nombreuses solutions commerciales de filtrage est basé sur l'utilisation de ces listes, complétées par une panoplie d'autres méthodes de filtrage, parfois obscures ou gardées secrètes. La multiplicité et la complexité des méthodes devient un argument de vente.

Kolcz [149] énumère les difficultés du filtrage individualisé chez un ESP (Email Service Provider) hébergeant quelques millions de boîtes aux lettres et établi des limites de faisabilité se basant sur des critères de coût et bénéfice (modèle économique). Kolcz estime que cette option est à privilégier par rapport à un filtrage collectif lorsque les spams sont dominants dans le flot et qu'une quantité encore importante de spams passe au travers d'un filtrage collectif.

CanIt[2] et *j-chkmail*[3] sont deux filtres anti-spam libres avec un classificateur "Bayésien Naïf" intégré, qui distribuent les informations d'apprentissage. Néanmoins, le mode de collecte et constitution des ensembles d'exemples utilisés pour l'apprentissage diffère.

CanIt collecte des signatures (liste de termes) des messages reçus par des utilisateurs (volontaires) du filtre [244] et les valide avant de les intégrer dans une base commune qui sera distribuée à l'ensemble des utilisateurs du filtre. En 2006, dans un forum de discussion, l'auteur de ce logiciel indiquait que la base de signatures contenait 6.548.626 termes extraits de 446.740 spams et 181.182 hams[4]. L'idée derrière cette procédure est que l'ensemble de messages collectés est représentatifs du flot de messages à filtrer et, comme dans le cas d'utilisation de corpus publiques de messages, le critère de classement ne dépend pas du destinataire.

[2] *CanIt* - `http://www.roaringpenguin.com`
[3] *j-chkmail* - `http://www.j-chkmail.org`
[4] `http://objectmix.com/sendmail/207206-spamassassin-db.html`

La base de termes distribuée par *j-chkmail* est destinée plutot au filtrage spécifique dans une population du type campus universitaire. Le corpus d'apprentissage de *j-chkmail* est constitué, pour les *hams*, de messages en provenance de listes de diffusion publiques caractéristiques de cette communauté et de messages donnés par des volontaires et pour les *spams*, de messages reçus par l'auteur ou dans des pièges à spam. En mai 2011, cette base était constituée de 934468 termes extraits de 72819 spams et 63395 hams. De ces 934468 termes, seulement 201817 termes sont partagés par les deux classes. L'idée derrière cette procédure est que dans le cas d'un classificateur génératif (Bayésien Naïf) le processus de filtrage revient, en fait, à comparer le vocabulaire d'un message avec ceux des classes et de choisir la classe dont le vocabulaire ressemble le plus.

Les démarches de ces deux filtres sont assez "étonnantes" : d'une part, rien ne permet d'affirmer que les messages collectés soient représentatifs du flot de messages à classer et, d'autre part, pour des raisons de respect de la vie privée, il n'existe pas d'évaluation sérieuse de leur efficacité. Le manque d'évaluation d'efficacité, en continu, fait que, malgré les constantes mises à jour, ces filtres fonctionnent en boucle ouverte et, en effet, le nombre important de termes suggère qu'ils soient sur-ajustés.

Néanmoins, ces filtres fonctionnent assez bien, selon l'impression subjective que l'on peut obtenir des utilisateurs de ces filtres. L'explication la plus probable est que les vocabulaires des classes sont assez disjoints, comme nous verrons dans le chapitre 12. Cela fait que la probabilité à priori des classes et la probabilité conditionnelle des termes sachant la classe ont peu d'importance dans le classement : la connaissance des termes présents dans chaque classe suffit pour obtenir des résultats satisfaisants.

Segal [233] propose l'apprentissage global d'un classificateur Bayésien Naïf avec la possibilité d'utilisation d'informations individualisées au niveau des termes. Au moment du classement d'un message on utilise, pour chaque terme, l'information globale, sauf si l'information individuelle existe avec un niveau de confiance supérieur. Cette méthode est mise en pratique dans le filtre *DSPAM*[5]. Ce filtre, utilisé à Rice University, contient 750K termes dans la base commune et, en moyenne, un peu moins de 20K termes dans la base de chaque utilisateur. Cela fait que, en moyenne, chaque utilisateur a, à sa disposition une base d'environ 770K termes. Le nombre total de termes de tous les utilisateurs, sachant qu'il y a 15.000, est de l'ordre de 300M + 750K[6]. Il est à remarquer que ce mélange de données obtenus à partir de sources différentes n'est envisageable que sous l'hypothèse d'indépendance statistique des termes.

Enfin, on remarque qu'un des objectifs des fournisseurs de solutions commerciales de filtrage de contenu est d'avoir un produit prêt à l'emploi (ou presque), quelque soit l'environnement d'utilisation. Pour atteindre ce but, un principe souvent utilisé est de baser le filtrage sur l'identification des spams et non pas le classement en deux catégories. Cette tactique dispense (presque) le besoin de collecte de messages légitimes mais exige une très bonne connaissance des spams en circulation, que l'on obtient grâce à des messages collectés par des nombreuses sondes, pièges à spam ou dénonciation par les utilisateurs du produit en question. Ces filtres sont très souvent mis à jour plusieurs fois par jour. Il est probable, mais pas démontré, qu'ils soient, en quelque sorte, sur-ajustés et que leur efficacité puisse se dégrader très rapidement en absence de mises à jour. Ces produits utilisent des heuristiques, à la fois complexes et artisanales, pour sélectionner les messages en provenance des pièges à spam qui seront utilisés pour le paramétrage des filtres [191].

Yih et al [249] [261] estiment, basés sur des analyses de contenu des messages de 200.000 utilisateurs volontaires de *hotmail.com*, qu'une quantité importante d'erreurs de filtrage proviennent des *Gray Mail*, des messages dont le classement varie d'un destinataire à l'autre. Ils supposent que ces messages sont, pour la plupart, des messages publicitaires et proposent de les identifier, par détection de campagnes publicitaires - un certain nombre de messages identiques (ou presque) envoyés dans un intervalle de temps assez court (I-Match [50] [150]) - et de traiter ces messages séparément. Le principe est intéressant mais il n'a pas été par démontré ni que les résultats puissent être transposées à d'autres types de communauté, ni qu'il soit applicable dans des communautés dont le niveau de trafic n'est pas aussi important que *hotmail.com*.

[5] *DSPAM* - `http://dspam.sourceforge.net`

[6] Ces informations résultent d'une correspondance privée avec Kenneth Marshall de Rice University

7.5 Les modes d'apprentissage courants

Nous pouvons identifier au moins quatre modes d'apprentissage dans les applications de filtrage de spam, que ce soient dans les solutions commerciales ou diffusés avec licence libre ou des propositions issues de la recherche.

Apprentissage sur les erreurs - Il s'agit d'un mode d'apprentissage où le destinataire retourne les informations sur les erreurs de classement détectées. Généralement, il ne détecte pas toutes les erreurs et ne retourne pas les erreurs de la classe *hams*, sauf lorsqu'il s'agit de messages anodins (pour des questions de confidentialité).

Apprentissage unilatéral (one side learning) - Cette approche se justifie par la difficulté de collecte d'exemples de la classe *ham*. Le principe de fonctionnement des filtres n'est pas le classement des messages en deux catégories mais l'identification des *spams*, tout le reste étant considéré comme étant des *hams*. Il s'agit d'une approche courante dans les solutions commerciales. Les spams proviennent, la plupart, des pièges à spam et, parfois, des messages retournés par des utilisateurs des solutions.

Apprentissage sur des corpus synthétiques - Cette approche consiste à collecter des exemples à partir de listes de discussion e nouvelles pour les *hams* e de messages des pots de miel pour les spams.

Auto-apprentissage - Il s'agit d'un mode d'apprentissage automatique, souvent dans des filtres comportant deux (ou plus) classificateurs - les messages soumis à l'apprentissage d'un classificateur statistique sont ceux classés par l'autre classificateur, pré-construit. Le classificateur construit de cette façon a souvent tendance à dériver progressivement, au fur et à mesure que les erreurs d'un classificateur se propagent à l'autre.

Ces modes d'apprentissage (ou plutôt paramétrage, selon le cas) ont souvent des inconvénients pratiques qui impactent leur efficacité ou leur utilité.

Ils visent surtout à dépendre, le moins possible, de l'intervention du le destinataire. Dans tous les cas, il s'agit d'un fonctionnement en boucle (presque) ouverte. Il n'y a donc aucune garantie que le modèle interne des classificateurs soit représentatif du flot à classer.

Du mode de fonctionnement en boucle ouverte et non intervention du destinataire des messages découle l'impossibilité d'une évaluation objective de l'efficacité de classement, autre que la satisfaction ressentie par les utilisateurs de la messagerie.

Le dernier inconvénient concerne la pertinence des exemples utilisés pour l'apprentissage, qui ne sont jamais choisis par le classificateur, mais selon des critères probablement pas optimaux. A cause de la course vers la perfection et le filtrage parfait, on peut se demander si les classificateurs construits de cette façon ne sont pas sur-ajustés.

7.6 Discussion et Conclusions

Dans ce chapitre nous avons pu identifier trois types de communautés ayant des caractéristiques différentes. Ces différences sont telles que la stratégie de filtrage optimale peut varier selon le type de communauté.

Les fournisseurs de solutions commerciales cherchent a créer des produits passe partout. Les solutions typiques sont des solutions utilisant tout d'abord des listes noires - ce sont des solutions qui rejettent les messages par leur provenance : les origines douteuses. Ce sont des solutions qui ne dépendent pas des caractéristiques de la communauté d'appartenance du destinataire. L'efficacité de ces solutions ne dépassent pas, en général, un taux de détection de l'ordre de 70 à 80 %. Ces solutions sont, en général, complétées par des solutions de filtrage de contenu. Ces solutions de

filtrage de contenu sont, le plus souvent complexes et intègrent rarement des retours d'information des destinataires.

Il s'agit de solutions fonctionnant en boucle (presque) ouverte à cause du faible niveau de retour d'information pris en compte pour la mise à jour du système de filtrage.

Très peu sont les travaux publiés concernant le classement mutualisé de message électroniques. Ils s'appliquent, en général, à un environnement d'entreprise ou alors ils ne tiennent pas compte du type de communauté.

Une des voies exploitées consiste à utiliser des classificateurs mixtes, utilisant des informations à la fois globales et spécifiques au destinataire [233]. Cette voie ne semble intéressante que dans les contextes où le nombre d'utilisateurs de la messagerie est limité ou quand le module de filtrage est proche de la boite aux lettres de l'utilisateur, à cause du besoin de gestion des préférences de chaque utilisateur.

Yih et Chang [249] [261] soulèvent le problème des messages publicitaires, problème qui n'est pas spécifique au filtrage mutualisé, mais qui prend de l'importance dans ce contexte, à cause des différences d'appréciation par les destinataires.

Très peu de résultats publiés sur le filtrage de spams tiennent compte ou mentionnent les caractéristiques particulières des corpus de messages utilisés, autres que la quantité par classe. Pour des questions de confidentialité, les messages légitimes de test provenaient assez souvent de listes de discussion publiques ou des boîtes aux lettres de plusieurs individus [9], [220]. Drucker et al [92] ont utilisé des messages provenant d'une seule boîte aux lettres mais, surement pour les mêmes raisons, les messages utilisés n'ont pas été rendus publiques. Les messages distribués par listes de discussion ne constituent qu'une partie des messages reçus par un destinataire quelconque et présentent, en général, une diversité assez faible et ne sont donc pas représentatifs d'un flot réel de messages.

Le corpus TREC Spam 2005 [64] [65] a été le premier corpus public de taille importante, constitué à partir de boîtes aux lettres d'individus identifiables d'une même communauté : il s'agissait du contenu de la messagerie de la société Enron, tombés dans le domaine public lors de la faillite de l'entreprise. Malgré l'intérêt de ce corpus, les messages couvrent une période autour de la banqueroute et reflètent une situation exceptionnelle et non pas pas une situation de fonctionnement stable dans une entreprise. Le corpus TREC Spam 2007 porte une amélioration dans le sens où les messages des deux classes ont été reçus en même temps sur le même serveur de messagerie. Néanmoins, d'une part il s'agit encore d'utilisateurs fictifs et, d'autre part, les messages légitimes ont été distribués par des listes de diffusion auxquelles ces utilisateurs, fictifs, ont été abonnés.

La réflexion menée dans ce chapitre suggère un renforcement du fonctionnement de la boucle de retour d'information avec de l'apprentissage actif.

CHAPITRE 8

Caractéristiques spatiotemporelles d'un flot de messages

La statistique est la première des sciences inexactes.

Edmond et Jules de Goncourt

8.1 Introduction

Le but de l'apprentissage d'un classificateur est la construction d'un modèle (ou une fonction) à partir d'un ensemble d'exemples *représentatifs* de la population d'où seront extraits les objets à traiter : les exemples résultent d'un tirage aléatoire *i.i.d.* de la population d'objets.

Dans un cas général de classement d'objets il est possible que les exemples ne soient pas significatifs de la population, et cela pour plusieurs raisons. Dans le contexte qui nous concerne, le classement mutualisé de messages électroniques, on peut avancer deux raisons : la constitution d'un ensemble d'échantillons significatifs du flot de messages peut ne pas être une tâche triviale et même si cela était possible, il peut y avoir une dérive temporelle.

Dans ce chapitre, nous menons une réflexion sur les causes de ces décalages, et la façon dont elle se manifeste. Dans le Chapitre 9, nous proposons une solution basée sur une architecture d'apprentissage actif en ligne.

8.2 Décalage et dérive

Un des challenges des techniques d'apprentissage artificiel est le décalage pouvant exister entre la situation d'apprentissage d'un classificateur et la situation de fonctionnement réel. L'hypothèse habituelle implicite dans des développements de méthodes d'apprentissage artificiel est que les exemples utilisés pour l'apprentissage et ceux utilisés pour les tests ont été générés à partir d'une même distribution de probabilité, inconnue mais constante, (voir, par exemple, Schölkopf et Smola [225, p. 8] ou Vapnik [256, p. x]), *c.à.d.* , ils résultent tous d'un échantillonnage aléatoire indépendant et identiquement distribué d'une même population. Cela suppose, accessoirement, que l'efficacité de classement est optimale lorsque cette hypothèse est vérifiée, ce qui n'est pas toujours le cas. Par exemple, le développement d'une application de classement de documents textuels développée et validée dans un pays anglophone (*e.g.* les États Unis) peut ne pas présenter pas la même efficacité dans un pays francophone (*e.g.* la France) puisque la diversité linguistique est différente dans chacun de ces deux pays. Dans cet exemple, ce n'est même pas l'adéquation

des données d'apprentissage qui est en cause, mais l'adéquation de l'algorithme de classement ou d'apprentissage.

Nous allons nous intéresser plutôt aux décalages spécifiques à l'application de filtrage de spam et en particulier ceux qui concernent l'utilisation partagée d'un filtre par plusieurs utilisateurs.

L'expression - *Dérive spatiotemporelle dans un flot de messages* - comprend deux formes de décalage entre l'ensemble d'apprentissage et le flot de messages à classer :

- **Le décalage spatial** qui nous intéresse est spécifique au filtrage partagé par plusieurs destinataires : le modèle appris par un classificateur correspond au flot global, addition des flots de tous les destinataires et non pas au flot de chaque destinataire.
- **La dérive temporelle** contient deux composantes : une dérive due au fait que le flot de messages n'est pas un processus stationnaire et un décalage dû à l'interaction avec les destinataires des messages, en particulier les retards dan la boucle de retour d'information. Dans ce chapitre nous examinons seulement la première composante. L'effet des retards de la boucle sera examiné empiriquement dans le Chapitre 11.

Dans la bibliographie, l'utilisation des termes *décalage* et *dérive* (*shift* et *drift*, en anglais) peut, parfois, prêter confusion. Un *décalage* a une connotation plutôt statique : la différence entre les distributions associées à deux processus différents, tandis qu'une *dérive* concerne plutôt la différence, à deux moments différents, entre les distributions d'un même processus. Si bien que dans les deux cas, il s'agit de la différence entre les ensembles d'apprentissage et de test, les conséquences et le traitement ne sont pas les mêmes. Dans le cas statique (décalage), on peut imaginer que *"l'écart"* puisse être évalué une fois pour toutes et utilisé pour corriger le classement, ce qui n'est pas possible dans le cas de la dérive temporelle puisque cet *"l'écart"* doit être évalué et corrigé en permanence.

Aussi, les expressions *Concept Drift* [262] [253] [158] ou *Population Drift* [143] sont parfois utilisées pour désigner un décalage dans la distribution des objets. Parfois on utilise *Concept Drift* et *Concept Shift* pour désigner des changements graduels ou brusques [177].

Plusieurs peuvent être les causes d'un décalage entre les exemples d'apprentissage et de teste, mais on peut les regrouper selon leur impact. Storkey [246] énumère quelques regroupements possibles (non spécifiques à la problématique du filtrage de spam) :

1. **Décalage de la probabilité à priori** - (*Prior Probability Shift*) Ce changement est assez courant. Les algorithmes génératifs (*e.g.* , Bayésien Naïf) considèrent que les messages sont générés selon un modèle du type $P(\mathbf{x}|y)\,.\,P(y)$, où y est la classe et $\mathbf{x}$ est un message. Le classement se fait à partir d'une estimation de $P(y|\mathbf{x})$ effectuée à l'aide de la règle de Bayes. Si $P(y)$ change, la règle de décision doit changer, elle aussi.
2. **Échantillonnage avec tendance** - (*Sample Selection Bias*) Ce type de décalage apparaît lorsque la constitution de l'ensemble d'exemples ne résulte pas d'un échantillonnage *i.i.d.* de la population et que certaines régions de l'espace de représentation des objets ne sont pas représentées dans l'ensemble d'apprentissage. Dans le cas de classement des spams, par exemple, cela peut être l'utilisation uniquement des messages classés en erreur, la non utilisation de messages sensibles ou encore l'inclusion plutôt des spams que des hams.
3. **Décalage simple des attributs** - (*Simple Covariate Shift*) il s'agit d'un changement dans la probabilité à priori des attributs $P(\mathbf{x})$, alors que la probabilité à posteriori des classes $P(y\,|\,\mathbf{x})$ reste constante. Storkey [246, p. 8] défend que ce cas n'affecte pas le modèle $P(y\,|\,\mathbf{x}^*)$, alors que, par exemple, Shimodaira [241] Sugiyama et al [248] ou Bickel et al [20] considèrent qu'il s'agit d'une variante de l'échantillonage avec tendance.
4. **Classes non équilibrées** - (*Unbalanced Data*) Il s'agit de problèmes avec des exemples rares qui n'apparaissent pas dans l'ensemble d'exemples d'apprentissage. On peut le comprendre (mais pas le traiter) comme une variante d'échantillonage avec tendance.
5. **Décalage de domaine** - (*Domain Shift*) Il s'agit d'une évolution dans la définition des classes.
6. **Décalage dans une composante source** - (*Source Component Shift*) lorsque les échantillons proviennent d'un mélange de différentes sources indépendantes (voir Chapitre 14),

chacune avec sa propre distribution de probabilités et son coefficient de mélange associé, il est courant que chaque source puisse évoluer indépendamment des autres. La difficulté de ce cas, en ce qui concerne le filtrage de spams, est que ni le nombre de sources n'est pas facilement identifiable, ni ses paramètres.

Ces cas de décalage impactent différemment les algorithmes de classement, selon qu'ils ont une interprétation plutôt géométrique (*e.g.* les SVM et le Perceptron) ou probabiliste (*e.g.* le Bayésien Naïf ou Régression Logistique).

Cette typologie n'est ni unique ni exhaustive. Vu d'un point de vue probabiliste, les conséquences des décalages se présentent sur : la probabilité à priori des classes $P(y)$, la probabilité des exemples conditionnée à la classe $P(\mathbf{x}|y), i = 1, \ldots, N$, ou la probabilité à posteriori des classes $P(y|\mathbf{x})$ [158].

8.2.1 La dérive temporelle

Il est généralement admis que la génération de messages électroniques, en particulier le spam, est un processus non-stationnaire [100] [60] [69] puisque les caractéristiques des messages évoluent dans le temps. On considère que l'efficacité de filtrage est optimal lorsque les exemples et les messages à classer sont générés à partir d'une même distribution de probabilité. Cette contrainte n'est jamais satisfaite dans un processus non-stationnaire puisque l'apprentissage se fait avec des messages du passé alors que les messages à classer n'existeront que dans le futur. La solution intuitive, et naïve, pour minimiser l'écart entre ces deux ensembles est de renouveler l'apprentissage périodiquement ou en continu. Néanmoins, la difficulté de constitution et d'entretien d'un corpus d'exemples étiquetés pousse à l'utilisation d'exemples qui ne sont pas forcément récents et de ne les renouveler qu'occasionnellement lorsqu'une baisse d'efficacité de filtrage est constatée. En tout état de cause, le classificateur ne doit pas dépendre d'un retour d'information concernant chaque message classé.

Il est vraisemblable que la classe des messages légitimes évolue naturellement alors que dans la classe des *spams* les changements sont intentionnels. On peut aussi admettre intuitivement que les deux classes évoluent de façon indépendante, que les *spams* évoluent plus rapidement que les *hams* et que le renouvellement des exemples peut se faire indépendamment dans les deux classes. Cette hypothèse, justifiée intuitivement ci-après, sera examinée expérimentalement dans le Chapitre 10.

La Figure 2.4 du chapitre 2 montre que la répartition des messages dans les classes (probabilité à priori des classes) évolue dans le temps, à court et à long terme. Ce changement quantitatif est directement observable mais il est accompagné de modifications plus profondes. Regardons ce qui se passe dans les deux classes.

Tout d'abord, la classe des messages légitimes. On peut raisonnablement énumérer les types de changements que l'on retrouve dans cette classe. Ces messages sont écrits par des personnes qui n'ont pas des raisons, à priori, pour modifier leurs habitudes rédactionnelles (style, vocabulaire) mais qui peuvent, de temps en temps, changer la présentation (mise en forme, couleurs de police, ...) suite à un changement ou mise à jour du logiciel de messagerie ou tout simplement pour échapper à la monotonie. Aussi, à l'exception de certaines catégories professionnelles telles la communication ou la vente, chaque personne a son réseau de correspondants que l'on peut considérer comme raisonnablement stable. On peut évidement rejoindre ou quitter des groupes de discussion. Mais, globalement, on peut considérer que, mis à part des événements exceptionnels tel un changement de domaine d'activité, les caractéristiques des messages reçus par quelqu'un évoluent dans le temps, mais ne subissent pas des modifications conséquentes et fréquentes. Ce sont des évolutions que l'on peut classer comme naturelles. Si on se place à un niveau plus élevé, celui d'un groupe d'individus, on peut émettre l'hypothèse que les changements individuels se diluent dans le flot de messages regroupés et peuvent deviennent imperceptibles, ou alors ne se manifester que comme une augmentation du bruit de fond, mesurable par la variance d'un certain paramètre tel un score de classement.

Les changements dans la classe spam ne sont pas de la même nature. La génération de ces messages peut être vue comme la confrontation entre deux opposants [100] [179] [180] : le spammeur et le service de filtrage de spams, chacun adaptant son stratégie en fonction des actions de l'autre, une *"course aux armements"*. Il s'agit de changements intentionnels, avec une fréquence suffisamment élevée pour pouvoir s'adapter aux changements de l'adversaire.

Les messages de cette classe concernent un nombre limité et pas très important de sujets (pornographie, médicaments, ...). Selon Spamhaus[1], 80 % des spams en circulation sont générés par seulement 100 spammeurs. Ce faible nombre d'expéditeurs fait que les oscillations dans leur activités impactent directement le nombre de spams en circulation [100] et la probabilité à priori de chaque classe. Ce sont des changements directement observables dans le trafic.

Les changements dans les caractéristiques cachées (tels le vocabulaire, les sujets de discussion, ou le style rédactionnel) sont plus difficiles à détecter et peuvent être confondues avec du bruit [262] : ces changements ne peuvent pas être déduits directement des paramètres mesurables.

L'évolution de la quantité de spam en circulation, avec une répartition par genre, a été constaté par Fawcett [100] ou Hulten et al [11]. Plusieurs éditeurs de logiciels de filtrage (Sophos, McAfee, Symantec, Brightmail, ...) présentent périodiquement des rapports ou des livres blancs avec ce type d'information, parfois même en temps réel sur leurs sites web.

Pu et Webb [204] ont étudié l'évolution des stratégies de création de spam sur une période de 3 ans (de janvier 2003 à décembre 2005) à partir des messages du corpus public SpamArchive [2]. Pour cela, ils ont utilisé les attributs définis par le logiciel SpamAssassin[3] et observé les dates d'apparition et disparition de ces attributs dans les messages. Cet étude est intéressante puisque l'observation porte non pas sur la quantité de messages en circulation, un indicateur du niveau d'activité des expéditeurs, mais sur la façon de construire les messages, ce qui relève plutôt de la stratégie des expéditeurs pour pouvoir traverser les filtres.

Sheng et al [240] ont étudié l'activité des sites de Phishing (Hameçonnage) et ont observé qu'une campagne de cette catégorie de spam ne dure que quelques heures alors que les sites web vers où les victimes sont attirés restent actifs plus longtemps. Dans ces conditions, les méthodes heuristiques et les filtres statistiques sont plus efficaces que les listes noires de URLs, qui ne sont pas mises à jour suffisamment vite.

Delany et al [86] ont publié quelques résultats d'évaluation de dérive des performances due au manque de mise à jour du modèle appris, mais obtenus avec un ensemble de teste dont la taille ne semble pas significative (quatre ensembles de seulement 1000 messages ayant, chacun le même nombre de *hams* et de *spams*) et avec une méthodologie semblant plutôt destinée à la validation de l'algorithme en de classement proposé.

La plupart de ces travaux sont spécifiques à la classe *spam* et, à notre avis, ne vont pas assez loin, dans la recherche d'une meilleure compréhension des caractéristiques temporelles des flots de messages électroniques.

Avec Gordon V. Cormack [71] [81] nous avons étudié l'effet de l'âge des exemples ainsi que de leur âge relatif sur l'efficacité d'un filtre utilisant un algorithme de régression logistique. Nous avons montré que partie de la dérive constaté dans l'efficacité de filtrage est due à la présence dominante de références temporelles [71], des références qui ne sont pas liées à la classe de chaque message. mais qui font en sorte que, si les messages de chaque classe utilisés lors de l'apprentissage n'ont pas le même âge, le filtre fini par apprendre à classer par âge et non pas par catégorie de message. Cette remarque est intéressante puisqu'elle peut remettre en cause les résultats de travaux validés à l'aide de corpus de messages synthétiques et dont les âges des messages des deux classes ne sont pas comparables, ou qui ont été validés utilisant des méthodes plutôt destinées aux processus de classement hors-ligne.

Avec les expérimentations présentées dans le Chapitre 10, nous avons pu mieux identifier les caractéristiques de la dérive de chaque classe et l'impact sur l'efficacité de classement.

[1] *SpamHaus ROKSO* : `http://www.spamhaus.org/rokso/index.lasso`
[2] *SpamArchive* : `ftp://spamarchive.org/pub/archives`
[3] SpamAssassin : `http://spamassassin.apache.org`

8.2.2 Le décalage spatial

Le décalage spatial, dans le sens qui nous intéresse, représente de la *différence* pouvant exister entre l'ensemble d'apprentissage et l'ensemble de classement réel mais aussi entre plusieurs flots de messages : des utilisateurs différents, des groupes différents d'utilisateurs. Ce problème est, en quelque sorte, assez différent de celui de la dérive temporelle.

Intuitivement, on accepte que les boites aux lettres de deux personnes distinctes soient aussi "différentes". Mais on peut se demander si les boîtes aux lettres de deux personnes travaillant dans le même service ou dans des services différents sont aussi "différentes". En fait, nous cherchons à quantifier la différence entre deux boîtes aux lettres ou, au moins, être capables de les ordonner selon un critère à définir.

Du point de vue classement de messages, une notion pertinente de "distance" est celle qui exprime la différence entre les flots *telle que ressentie par le classificateur*. Le Chapitre 13 est un chapitre de perspectives et présente quelques réflexions sur des possibles méthodes pour évaluer la différence entre ensembles de messages.

Dans le contexte qui nous concerne, le décalage spatial joue un rôle important. Dans une communauté assez large, on telle une université, on peut intuitivement effectuer quelques hypothèses sur les types de message en circulation. On peut éventuellement obtenir des échantillons significatifs du flot de messages de quelques utilisateurs mais il est quasiment impossible de le faire pour l'ensemble des utilisateurs. Dans ce cas, il y a surement un décalage entre les échantillons obtenus, les flots des autres utilisateurs et le flot global de messages.

8.3 La diversité

La diversité des messages du flot est un point important dans le contexte de classement mutualisé : nous l'avons mentionné dans le Tableau 7.1 sous la rubrique "Ressemblance des boites aux lettres des utilisateurs" - la diversité est le manque de ressemblance, des points communs, entre les messages.

La boite aux lettres d'une personne est constitué d'un ensemble de messages pouvant être groupées par des critères communs tels l'appartenance à un groupe de travail, messages de service, amis, ... Les messages, à l'intérieur d'un même groupe, ont des points communs et une certaine ressemblance. A un niveau supérieur, on peut imaginer regrouper des boites aux lettres de personnes avec des activités semblables.

Intuitivement, on peut espérer que lorsqu'on combine les boites aux lettres de plusieurs personnes d'une communauté, plus leurs activités sont différentes, plus la boite combinée sera hétérogène. Ceci ne peut ne pas avoir un impact sur l'efficacité de classement.

8.4 L'apprentissage dans un flot non stationnaire

La dernière décennie a vu une explosion d'applications manipulant des flots de données évolutifs et faisant appel aux techniques d'apprentissage artificiel. Le déploiement d'Internet a largement contribué. Parmi ces applications, on trouve la fouille de données [110], la détection de changements, et des vitesses associées, dans un flot [2] [27]. Le classement en ligne et le clustering [3] [265] [258] sont des domaines qui ont des similarités avec le domaine de classement de messages électroniques. Les résultats de recherche concernant le filtrage de spam, tenant compte du caractère non stationnaire du flot de messages sont plutôt rares.

La caractéristique première d'un flot non stationnaire est, bien entendu, la constante évolution de ses caractéristiques statistiques, faisant que le classificateur doit être reconstruit en permanence pour s'adapter. Pour ce faire, il y a deux deux familles d'approches - l'apprentissage sur les exemples dans une fenêtre temporelle glissante et l'apprentissage incrémentale - puis une approche intermédiaire : les "mini-batches".

8.4.1 Apprentissage sur les exemples pris dans une fenêtre temporelle

C'est l'approche canonique ! Il s'agit de définir une fenêtre temporelle ayant une extrémité quelque part dans le passé et l'autre à l'instant présent. La fenêtre se déplace au fur et à mesure que le temps avance, intégrant des exemples nouveaux et supprimant des anciens qui ne sont plus pertinents. Cette approche apparait naturellement dans les classificateurs basés sur les exemples (que l'on appelle parfois *"lazy learners"*) et qui utilisent explicitement l'ensemble des exemples au moment du classement [262].

Si l'idée est simple, sa mise en place l'est moins.

Si le flot de messages était un processus stationnaire ou si son évolution était déterministe, l'avancement de la fenêtre temporelle serait prévisible : la vitesse des chacun des deux fronts (pas forcément identiques), et le nombre d'exemples récents à ajouter et des anciens à supprimer.

Rien ne dit que les deux fronts, avant et arrière, de la fenêtre temporelle avancent à la même vitesse : la taille de la fenêtre n'est donc pas constante et le nombre d'exemples ajoutés diffère du nombre d'exemples supprimés. L'avancement du front avant de la fenêtre est trivial et imposé par l'avancement du temps - ce n'est pas le cas du front arrière. La détermination de l'avancement adéquat du front arrière relève souvent d'un traitement lourd pas forcément compatible avec les contraintes de traitement en temps réel souhaité dans le problème de classement en ligne de messages électroniques.

Une autre difficulté pratique est la place mémoire nécessaire pour enregistrer les exemples de la fenêtre glissante. En principe, tous les nouveaux exemples doivent être utilisés pour mettre à jour les modèles mais il est rare qu'ils soient tous disponibles, avec le classement correct associé, pour être ré-injectés dans la chaîne d'apprentissage.

Ces difficultés expliquent, probablement, le nombre réduit de résultats de recherche sur des filtres anti-spam basés sur ce principe, souvent spécifiques à l'algorithme de classement.

Avec cette approche, Cunningham [80] utilise un classificateur kNN (k voisins les plus proches) et refait l'apprentissage dès que le taux d'erreur dépasse un certain seuil. Fernandes-Riverola et al [102] se basent sur une méthode de sélection d'attributs pour choisir les exemples à supprimer ou à garder et pour re-évaluer la taille de la fenêtre temporelle. Hsiao [129] évalue les changement à l'intérieur de clusters (nuages) d'exemples pour déclencher des mises à jour de l'apprentissage. Delany et al [86] utilise un classificateur basé sur des exemples (lasy learner) : seulement les messages avec un classement erroné sont ajoutés à l'ensemble d'exemples et un re-apprentissage complet est relancé périodiquement pour pouvoir supprimer les exemples anciens devenus peu pertinents.

On remarque, néanmoins, que les résultats présentés dans ces publications ont été obtenus avec des corpus d'apprentissage et de test de taille assez limitée (quelques centaines/milliers d'exemples) qui ne nous semble pas représenter des situations réelles de filtrage de mail. On peut s'interroger sur la capacité de traitement de messages à une échelle plus importante.

L'avantage souvent mise en valeur par les adeptes de cette approche est la capacité de détection et adaptation aux changements locaux internes à chaque classe.

8.4.2 Apprentissage incrémental

La deuxième approche consiste à utiliser chaque nouveau exemple pour mettre à jour de façon incrémentale les modèles appris par le classificateur. Dans cette approche, les exemples sont soumis à l'apprentissage, dans l'ordre chronologique (donc, possiblement en temps réel), et supprimés immédiatement après. Le but premier est de ne pas pas conserver l'ensemble des exemples passés et d'éviter la complexité liée à la gestion de la fenêtre temporelle.

Indépendamment des détails liés à l'implémentation, l'idée de base est que la présentation de nouveaux exemples fait que le modèle construit de façon incrémentale converge asymptotiquement vers le modèle réel du trafic. Bien entendu, l'apprentissage doit intégrer un mécanisme permettant d'*oublier* ou de diminuer progressivement l'influence des anciens exemples au fur et à mesure de leur vieillissement.

L'apprentissage incrémental n'est pas adapté à tout type d'algorithme de classement. Un classificateur Bayésien Naïf, par exemple, comptabilise, pour chaque attribut, le nombre d'exemples où les attributs apparaissent. La relation entre les attributs et chaque exemple n'existant plus dans le modèle créé lors de l'apprentissage, le mécanisme d'oubli des exemples les plus anciens ne peut pas être mis en place, sauf si l'on garde les informations individualisées de chaque exemple mais, dans ce cas, ça revient à utiliser le modèle de fenêtre temporelle glissante.

Comme pour l'approche précédente, il faut que la fréquence de présentation des exemples pour apprentissage, ainsi que les différents retards soient compatibles avec la vitesse de changements observés dans le flot de messages.

Goodman et Yih [115] utilisent un algorithme de descente séquentielle du gradient pour l'apprentissage d'un classificateur à régression logistique, les messages étant présentés dans l'ordre chronologique. Cette méthode d'apprentissage ressemble celle d'un Perceptron [155] sauf que la convergence est atteinte par la présentation des exemples au fur et à mesure de leur occurrence et non pas par présentation en boucle des exemples mal classés. Les exemples sont utilisés une seule fois et oubliés ensuite. L'inconvénient de cette méthode est que l'apprentissage est faite avec tous les messages classés. Les résultats reportés sont très bons, meilleurs que ceux obtenus avec les classificateurs génératifs largement diffusés et confirmés pendant TREC Spam Track 2007 [57] [58]. L'algorithme d'apprentissage proposé par Goodman et Yih est, en effet, un cas particulier d'*approximation stochastique*, avec vitesse d'apprentissage constante.

Certains filtres libres "dits bayésiens" tels *Bogofilter*[4] enregistrent la dernière fois que chaque attribut a été vu dans une opération de classement et suppriment, périodiquement, ceux qui n'ont pas été vus depuis un certain temps. Vu que dans cette approche on décompte seulement les attributs et non pas les exemples, le modèle diverge progressivement du contenu réel du flot et doit être ré-initialisé de temps en temps.

8.4.3 L'apprentissage en *mini-batches*

Il s'agit d'une approche intermédiaire entre les deux précédentes, dont le but est plutôt pratique et qui vise surtout une optimisation des ressources matérielles. Il s'agit d'une approximation où l'on utilise, en général, une fenêtre temporelle de taille fixe et l'on refait l'apprentissage à des intervalles fixes ou lorsque le taux d'erreur dépasse un certain seuil.

Cunningham [80], par exemple, refait l'apprentissage d'un classificateur "lazy learner" lorsque le taux d'erreur dépasse un certain seuil.

Cette approche est aussi souvent choisie dans les classificateurs bayésiens naïfs tels j-chkmail[5], où les messages de la classe spam sont actualisés chaque jour.

L'apprentissage en *mini-batches* est aussi choisie lorsque la mise à jour du classificateur est trop coûteuse. Sculley [229] [226] propose ROSVM (Relaxed Online Support Vector Machine) pour le filtrage de spam. Il s'agit d'une version de SVM dont l'apprentissage s'effectue par paquets de nouveaux exemples, mais avec relaxation de certains paramètres, comme par exemple, apprentissage uniquement sur des erreurs, limitation dans le nombre d'itérations ou limitant l'apprentissage sur les p derniers exemples. Cette dernière contrainte fait que, en pratique, son approche relève d'une apprentissage sur une fenêtre temporelle glissante.

A. Bordes et al [29] propose *LASVM*, une approche alternative d'apprentissage en ligne d'un *SVM*, différente de celle de Sculley. Dans cette approche, la mise à jour du classificateur consiste à, exécuter deux processus *PROCESS* et *REPROCESS*. Le premier sert à ajouter des nouveaux exemples, si pertinents et le deuxième, à supprimer ceux qui ne sont pas des vecteurs de support et qui ont peu de chances de le devenir. A notre connaissance, il n'y a pas eu d'expérimentations de classement de messages eléctroniques utilisant *LASVM*.

[4]Bogofilter : http ://bogofilter.sourceforge.net/
[5]j-chkmail : `http://www.j-chkmail.org`

8.5 Conclusions

Vu que le processus de génération de messages électroniques n'est pas un processus stationnaire, le traitement de la dérive temporelle des caractéristiques statistiques qui résulte est considéré comme un besoin et plusieurs plusieurs propositions de solutions ont été publiées. Néanmoins, à notre connaissance peu de travaux ont été publiés sur l'évaluation des conséquences de cette dérive sur l'efficacité des filtres anti-spam.

CHAPITRE 9

Un filtre anti-spam avec apprentissage actif en ligne

La simplicité est la sophistication suprême.

Leonardo da Vinci

9.1 Classement et apprentissage actif en ligne

Après avoir étudié les briques de base nécessaires à la construction d'un système de classement de documents textuels, de tirer les enseignements des solutions actuelles de filtrage de spam et de leurs déficiences, nous présentons, dans ce chapitre, une architecture et des choix que nous estimons capables de répondre, à la fois aux besoins d'un système mutualisé de classement de messages dans une organisation et aussi à la démarche définie à l'introduction de ce document : un classificateur simple et efficace.

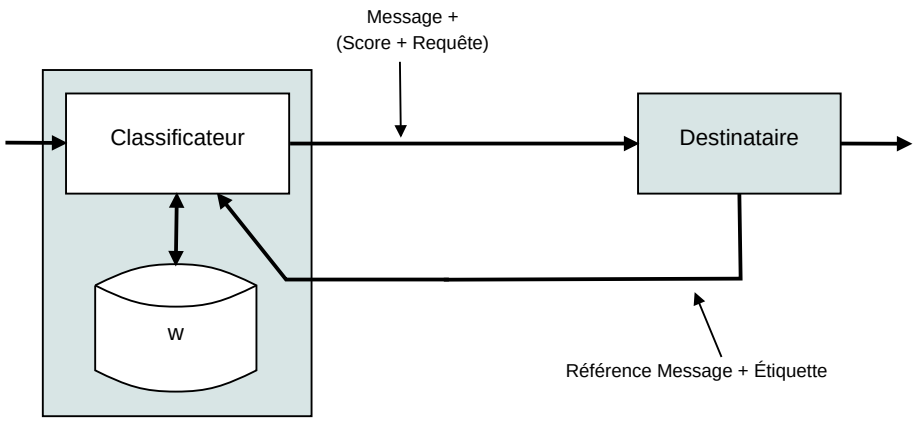


FIG. 9.1: Environnemnt de classement et apprentissage actif en ligne.

Une architecture de système de filtrage mutualisé (ou pas) avec retour d'information apparait naturellement dans la Figure 9.1. Les choix apparaissent aussi naturellement.

9.2 Représentation des messages

Le choix du niveau de décomposition - *mots* ou *caractères* - pour la représentation de messages ne peut pas se justifier ni de façon analytique, ni de façon objective. Il s'agit juste d'un choix justifié par des arguments subjectifs. Des arguments qui pourront ne pas être valables dans l'avenir ou dans d'autres contextes linguistiques.

Notre choix est d'utiliser des *4-grams de niveau caractère* extraits de la façon suivante : on considère une fenêtre textuelle de taille 4 caractères se déplaçant le long du message, caractère par caractère. Ainsi, la phrase "allons enfants de la patrie" va générer les 4-grams suivants : *"allo", "llon", "lons", "ons_", "ns_e", "s_en", "_enf", "enfa", "nfan", ...*

Plusieurs arguments justifient le choix niveau *caractère* et pas *mot* :

- **simplicité et précision** - l'heuristique d'extraction de *n-grams* de niveau caractère peut être définie de façon précise, tandis que l'extraction de *mots* suppose la définition d'un ensemble de caractères séparateurs. Certains de ces caractères seront ou pas des séparateurs, selon le contexte et la langue, par exemple le "." selon qu'il se trouve à la fin d'un mot (fin de mot et de phrase), entouré de chiffres (séparateur décimal) ou entouré de lettres (partie d'une sigle). Il n'y a pas d'heuristique simple et générale permettant de définir la règle d'extraction des mots ;
- **robustesse** - les n-grams de niveau caractère plus résistants aux attaques visant le processus de segmentation des messages. Une technique souvent utilisé les *"spammeurs"* est, par exemple, l'introduction de caractères séparateurs (espaces, virgules, caractères spéciaux, ...) entre les lettres des mots critiques (p. ex., *"V,I,A,G,R,A"*) de façon à perturber l'algorithme de segmentation des messages. Par ailleurs, ce niveau de décomposition est courante dans les outils de correction orthographique, puisqu'il permet de détecter et corriger les erreurs, considérées comme du bruit ;
- **regroupement par le sens** - l'utilisation des *n-grams* est une façon simple d'extraire la racine des mots permettant de regrouper, par exemple, des formes conjuguées. Dans ce cas de représentation de niveau *mot*, il faut faire appel à des opérations de *lemmatisation* et/ou *stemming*, avec des heuristiques non triviales et dépendantes de la langue utilisée.
- **dépendances inter mots** - l'extraction des *n-grams* au fil de l'eau permet de préserver, au moins partiellement, la relation de *succession des termes* dans le texte, ce qui n'est pas le cas lorsque l'unité est le mot.

La longueur, quatre, a été choisie de façon expérimentale, dans des expérimentations préliminaires, et constitue un compromis entre la complexité et la capacité d'extraction d'information : l'ordre 3 semble ne pas extraire suffisamment d'information et le niveau 5 génère des classificateurs exigent plus d'échantillons pour constituer un vocabulaire significatif.

Un argument contre l'utilisation de *mots* comme terme élémentaire de segmentation est l'environnement de fonctionnement hostile. En effet, dans la classe *ham* les messages sont bien rédigés alors que ce n'est pas le cas dans la classe *spam*. Des nombreuses astuces sont utilisées pour tromper les analyseurs lexicaux des classificateurs. Par exemple, le mot *"VIAGRA"* peut être écrit soit comme *"V I A G R A"* (insertion de caractères séparateurs entre les lettres) ou alors *"VI4@R4"* (remplacement de lettres par des caractères visuellement semblables). Le but étant de faire en sorte que le texte soit visuellement compréhensible par l'expéditeur mais pas par un analyseur lexical.

Enfin, ces choix sont à revoir dans les contextes de langue dominante dans le flot. Par exemple, la langue dominante peut appartenir à la famille des langues agglutinantes (turc, hongrois, ou japonais) ou être codée avec d'autres familles de caractères (langues asiatiques ou des pays de l'est). Malheureusement, nous n'avons pas d'échantillons de messages permettant de vérifier ces contextes.

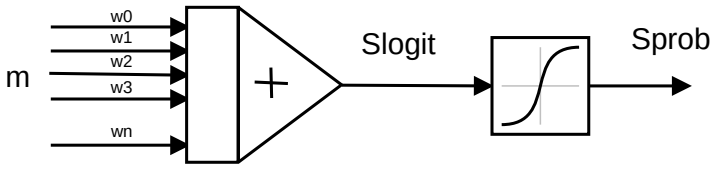


FIG. 9.2: SLDC - Un classificateur linear discriminant simple

9.3 SLDC - Simple Linear Discriminative Classifier

Nous avons choisi un algorithme du type discriminant linéaire, simple, décrit par la Figure 9.2. Du point de vue fonctionnel (en mode classement), il est similaire au *Perceptron*, avec la fonction de lien (link function) remplacée par une fonction sigmoïde. L'algorithme de classement s'inspire, en partie, de celui proposé par Goodman et Yih [115], avec une stratégie d'apprentissage différente (actif plutôt que supervisé) et des modifications dans la représentation des messages.

Le résultat du classificateur est une valeur de score, évaluée selon l'Équation 6.14, et présenté en deux échelles : *logit* $(-\infty, \infty)$ et *probabilité* $[0, 1]$. Ces scores sont donnés par :

$$\begin{aligned} S_{logit} &= -\langle \mathbf{w} \cdot \mathbf{m} \rangle \\ S_{prob} &= \frac{1}{1 + e^{-\langle \mathbf{w} \cdot \mathbf{m} \rangle}} \end{aligned} \tag{9.1}$$

avec $\mathbf{w}$ le vecteur de paramètres du classificateur et $\mathbf{m}$ la représentation du message à classer. La classe attribuée par le classificateur résulte d'un seuillage utilisant une valeur triviale de seuil :

$$\hat{y} = \begin{cases} 1 & \text{si } S_{logit} > 0 \text{ (spam)} \\ 0 & \text{si } S_{logit} \leq 0 \text{ (ham)} \end{cases} \tag{9.2}$$

Les deux scores S_{logit} et $\hat{y} = S_{prob}$ étant liés par la fonction sigmoïde, la valeur de seuil $S_{logit} = 0$, correspond à $S_{prob} = 1/2$.

Remarque 9.1 (Les échelles des scores). Les désignations S_{logit} et S_{prob} constituent, d'une certaine façon, un abus de langage et servent juste à établir un lien entre les échelles de valeur de ces deux présentations de score et ne correspondent pas à une estimation de probabilité à posteriori de la classe, comme on pourrait penser. La raison principale, dans le contexte présent, est que l'apprentissage actif ne résulte pas d'un échantillonnage *i.i.d.* de l'ensemble d'exemples.

9.3.1 Apprentissage

Notre choix d'apprentissage en ligne se fait par descente de gradient stochastique (voir Annexe C). La fonction de coût est l'erreur quadratique :

$$L_t = (y_t - \hat{y}_t)^2 \tag{9.3}$$

avec $y \in \{0, 1\}$ l'étiquette réelle du message et $0 \leq \hat{y} \leq 1$ est le score du message, estimé par le classificateur.

La mise à jour du vecteur de paramètres se fait selon :

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t + \eta_{t+1} \nabla_w L_t \\ &= \mathbf{w}_t + \eta_{t+1} (y_t - \hat{y}_t) \mathbf{m} \end{aligned} \tag{9.4}$$

Le taux d'apprentissage η_t suit une loi de récurrence définie par :

$$\eta_t = \frac{1 - \eta_\infty}{t} + \eta_\infty, \; 0 \leq \eta_\infty \ll 1 \tag{9.5}$$

La valeur résiduelle de η_∞ rend l'algorithme adaptatif lui permettant de suivre l'évolution d'un flot non stationnaire. Des valeurs typiques sont de l'ordre de 10^{-3} [18, p. 150]. De nos expérimentations préliminaires, nous avons trouvé qu'une plage raisonnable se situe entre 0.001 et 0.005.

En effet, l'utilisation d'une valeur constante et faible de η ferait juste ralentir la phase initiale d'accrochage, sans changement global significatif dans l'efficacité du classificateur. Néanmoins, il est intéressant de retenir le caractère variable de ce paramètre et de le rendre plus adaptatif en cas de changements abrupts dans les flots de messages. C'est un point que nous ne traitons pas dans cette thèse.

9.3.2 Apprentissage Actif

L'objectif de l'apprentissage actif est surtout le renforcement du fonctionnement du classificateur en boucle fermée. Il y a une différence fondamentale entre ce mode d'apprentissage et les options que l'on trouve dans les solutions de filtrage de spam : dans ce mode, c'est le classificateur qui prend l'initiative de choisir les exemples dont il souhaite connaitre la vraie classe associée, alors que dans les solutions de filtrage, ce choix est laissé à une entité extérieure, tel le destinataire, suite à une erreur de classement détectée. En apprentissage actif, le classificateur choisira les exemples que, dans son point de vue, sont les plus pertinents pour l'apprentissage.

Il s'agit d'un choix dans un flot, *c.à.d.* , il ne s'agit pas de choisir les exemples les plus pertinents d'un ensemble, mais décider au fil de l'eau, si un exemple en cours d'analyse est pertinent.

Le critère de décision est celui de marge fixe. Intuitivement, cela correspond sélectionner les exemples dont l'incertitude de classement se trouve au dessus d'un certain seuil. D'autres heuristiques ont été, *e.g.* [47], mais il n'est pas certains qu'elles soient plus efficaces. L'ensemble des messages pour lesquels le classificateur demande l'étiquette est défini par :

$$M = \{\mathbf{m} \ : \ |S_{prob}(\mathbf{m}) - 1/2| < Marge\} \tag{9.6}$$

Si, par exemple, la marge définie est de 0.35, le classificateur demandera l'étiquette de bon classement pour tous les messages dont le score S_{prob} est compris dans la place $[0.15, 0.85]$.

Cette heuristique pose un problème puisque les messages mal classés qui ne se situent pas à l'intérieur de la marge d'apprentissage actif ne feront pas objet de demande d'étiquette. Donc, nous avons considéré que le destinataire pourra prendre renseigner aussi renseigner l'information de classement correct pour ces messages.

Quatrième partie

Expérimentations

CHAPITRE 10

Caractéristiques temporelles empiriques des flots de messages

> On dit souvent qu'il faut expérimenter sans idée préconçue. Cela n'est pas possible ; non seulement ce serait rendre toute expérience stérile, mais on le voudrait qu'on ne le pourrait pas.
>
> *Henri Poincaré*, La Science et l'hypothèse (1908)

10.1 Introduction

Ce chapitre décrit une série d'expérimentations et de mesures effectuées dans un contexte de classement en "boucle ouverte". Ce contexte statique nous permet d'isoler et identifier les caractéristiques dynamiques d'un flot de messages.

On aurait pu utiliser l'expression "classement avec apprentissage hors ligne" ou "batch" au lieu de "boucle ouverte", expression courante plutôt en automatique. "Boucle ouverte" nous semble plus adaptée pour préciser qu'il s'agit bien d'un processus sans rétroaction : le classificateur utilisé a été construit au préalable et les informations devenues disponibles au fur et à mesure de l'avancement du temps ne sont utilisées pour actualiser le classificateur. L'absence de mises à jour nous permet d'observer, indirectement, l'évolution des caractéristiques temporelles du flot, à l'aide des indicateurs de classement.

Dans la première partie, nous examinons, par simple comptage, l'évolution de la répartition des messages par classe - mesure significative de la probabilité à priori des classes.

Dans la section suivante, nous utilisons un classificateur construit à partir des messages reçus par l'auteur pendant une période de deux mois pour classer quatre ensemble de messages reçus pendant les vingt mois suivants. Nous observons l'évolution des caractéristiques des flots de messages, à l'aide de méthodes issues de l'analyse de séries.

10.2 L'évolution de la répartition des messages par classe

La Figure (10.1) présente l'évolution, à long terme, du nombre de messages reçus par l'auteur, par jour et pour chacune des classes (ham et spam), pendant 32 mois (du 1er janvier 2008 au 31 août 2010). Le pas d'échantillonnage est la journée.

Le comportement du nombre de messages de la classe *ham* est approximativement stable, avec deux exceptions :

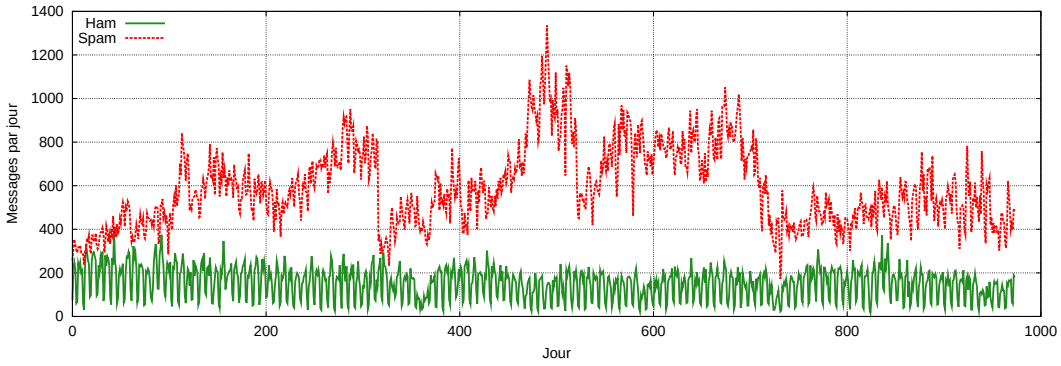


FIG. 10.1: Évolution à long terme de la répartition du nombre de hams et spams, par jour, reçus par l'auteur, entre le 1er janvier 2008 et 31 août 2010

– on remarque une claire périodicité hebdomadaire qui s'explique par la baisse de messages légitimes reçus pendant le weekend (2 jours sur cinq). Cette baisse s'explique par le fait qu'il s'agit d'une boîte aux lettres à usage professionnelle ;
– on peut aussi remarquer, des creux pendant les vacances d'été et de fin d'année.

Le flot de la classe *spam* ne présente n'est pas régulier et ne présente pas de périodicités évidentes et l'on peut juste remarquer quelques événements extraordinaires comme, par exemple, la fermeture du fournisseur d'accès McColo en novembre 2008[1] (vers le jour 320). L'absence de périodicités s'explique par le fait que l'ordonnancement de la distribution des spams est effectuée par des moyens automatisés, donc non soumis aux cycles d'activité humaines et professionnelles. Les événements et dérives dans le flot des spams sont plutôt liés aux aléas d'activité.

La Figure (10.2) montre la répartition des messages par classe, mais dans une échelle plus fine (comptage des messages à l'heure) et sur une durée d'un mois : juin 2008, un mois typique. Les quatre figures permettent de comparer le flot de messages de l'auteur et le flot global sur un des serveurs d'arrivée de l'École des Mines de Paris.

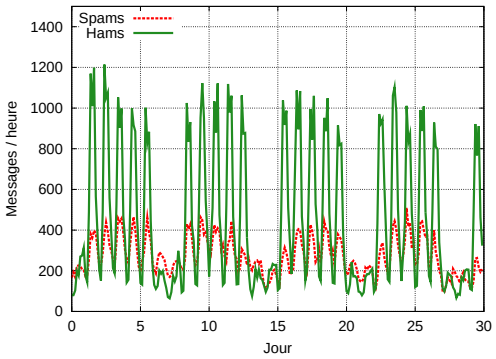
Outre le phénomène périodique hebdomadaire constaté dans l'observation à long terme, on constate un deuxième phénomène périodique journalier dû à la baisse d'activité pendant la nuit. Ce phénomène est plus accentué dans la classe ham que dans la classe spam, pour la même raison - la baisse d'activité humaine pendant la nuit. Ceci n'est pas repérable dans l'observation à long terme puisque le l'échantillonnage avait été fait avec des intervalles d'une journée entière.

L'autre observation pertinente concerne concerne le taux de spam bien plus important dans la boîte aux lettres de l'auteur que dans le flot global. Trois raisons permettent d'expliquer cela :
– des utilisateurs différents ne reçoivent pas la même quantité de spam : des adresses plus anciennes et plus divulguées ont plus de chance de recevoir plus de spam ;
– la messagerie de l'auteur inclut des adresses de service recevant habituellement beaucoup de *spam* (*postmaster*, *abuse*, ...) ;
– le flot global est soumis à un filtrage par *greylisting*. Cette méthode de filtrage protocolaire permet d'enlever beaucoup de spam sans toucher aux messages légitimes.

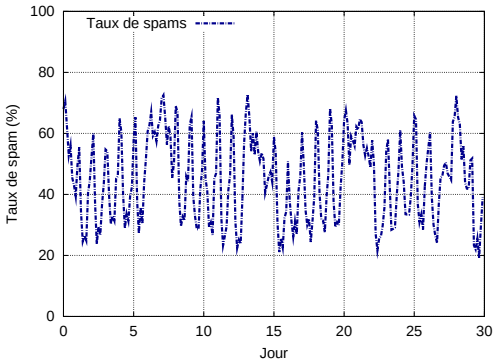
Ces observations posent deux problèmes distincts.

La variation observée dans la répartition des messages fait que la plage de variation de la probabilité à priori des classes est importante. On peut penser que l'efficacité de certains algorithmes utilisant explicitement ce paramètre (*e.g.* un classificateur Bayésien Naïf) ne peut pas être optimale. Or, on sait, par expérience, que ces classificateurs présentent des résultats qui sont toujours très bons, même si, à notre connaissance, aucun des algorithmes utilisés en pratique tient compte de cette variabilité temporelle. Nous essayons de donner une explication à cela dans le Chapitre 12.

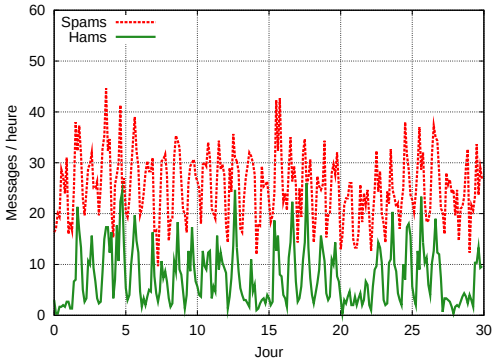
[1] McColo - Ce fournisseur d'accès internet était responsable de la distribution d'une fraction significative de spam. Voir, p.ex., `http://www.washingtonpost.com/wp-dyn/content/story/2008/11/12/ST2008111200662.html`



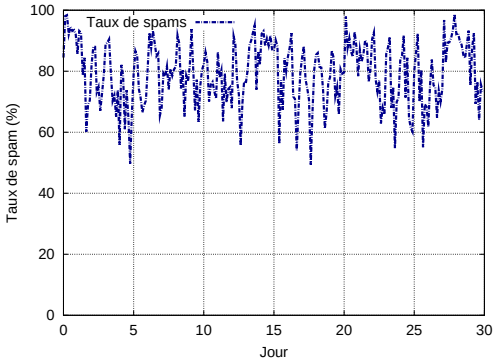
(a) Répartition des messages par classe pour l'ensemble des utilisateurs du domaine *mines-paristech.fr*



(b) Contribution spam dans le flot global du domaine *mines-paristech.fr*



(c) Répartition des messages par classe dans le flot de messages de l'auteur



(d) Contribution du spam dans les messages de l'auteur

FIG. 10.2: Répartition des messages par classe dans le flot global de messages du domaine *mines-paristech.fr* et dans les messages de l'auteur, pour le mois de juin 2008

Des événements périodiques apparaissent à plusieurs échelles - jour, semaine et année (les mois de vacances) - et plus particulièrement dans les messages légitimes. Les périodicités les plus rapides sont du même ordre de grandeur que les délais en jeu dans l'interaction avec les destinataires : la journée. Ces phénomènes seront examinés plus en détail dans la section suivante.

10.3 L'impact de l'âge et de l'âge relatif des exemples

Avec Gordon Cormack [71] [81], nous avons étudié empiriquement l'impact de l'âge, absolu ou relatif, des exemples dans l'efficacité de classement. Il ressort que l'efficacité de classement détériore effectivement, comme prévu, si l'apprentissage est faite avec des messages anciens. Néanmoins, l'impact le plus néfaste résulte de l'utilisation de messages dont l'âge n'est pas comparable, à cause des références temporelles, implicites ou pas, que l'on trouve dans les messages - *e.g.* les dates présents dans le message, les versions des logiciels ou encore les références à des événements largement diffusés. Leur présence systématique dans les messages peut faire qu'ils deviennent excessivement discriminants faisant que le classificateur apprenne à classer les messages par leur âge et non pas par le critère *ham*/*spam* souhaité.

10.4 Les caractéristiques temporelles d'un flot de messages

Cette section a pour but l'étude de l'idée généralement admise du caractère dynamique d'un flot de messages.

Dans l'expérimentation décrite dans cette section nous avons utilisé un classificateur, sans aucune mise à jour, pour classer des messages pendant une période suffisamment longue. Le but étant de faire apparaître, dans les indicateurs des résultats de classement, les conséquences de la dérive temporelle des caractéristiques d'un flot de messages.

Des séries temporelles ont été construites à partir des paramètres de classement de quatre flots différents de messages. Ces séries sont étudiées à l'aide des méthodes courantes d'analyse de séries de données : observation des séries brutes, vérification d'existence et identification d'une tendance déterministe, comparaison visuelle des variogrammes, ajustement d'un modèle linéaire avec ou sans saisonnalité (AR, MA, ARMA, ARIMA ou SARIMA) et analyse spectrale par périodogramme.

10.4.1 Introduction

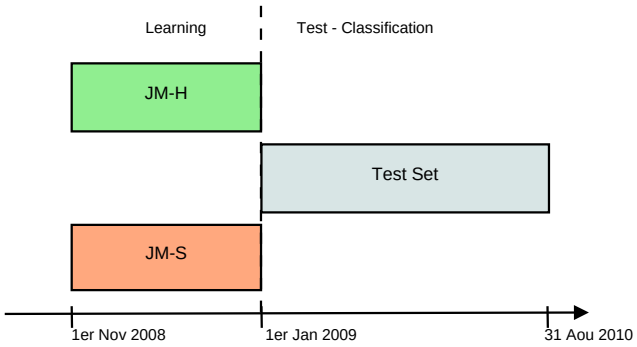


FIG. 10.3: Schéma temporel de l'expérimentation : les messages reçus pendant les mois de novembre et décembre 2008 sont utilisés pour la construction du classificateur, qui sera utilisé, sans être mis à jour, pour classer les messages de chaque ensemble de test.

Nous avons étudié les résultats de classement de quatre flots de messages :

1. **JM-H** et **JM-S** - les *hams* et *spams* reçus par l'auteur ;
2. **SB-S** - les *spams* reçus dans un piège à spam[2] géré par l'auteur et installé dans le domaine *ensmp.fr*. Il s'agit d'un serveur de messagerie acceptant des messages vers des adresses insérées de façon cachée dans quelques pages de certains serveurs web, qui ne sont pas forcément du même domaine ;
3. **BG-S** - les *spams* en provenance du corpus constitué par Bruce Guenter, disponible à `http://untroubled.org/spam/`. Ces messages proviennent de pièges à spam.

Les messages de chaque flot appartiennent tous à la même classe et couvrent une période de 600 jours : du 1er janvier 2009 au 31 août 2010. Les flots *JM-H* et *JM-S* correspondent aux deux classes de la boîte aux lettres de l'auteur.

Un classificateur a été construit à partir des messages reçus pendant les mois de novembre et décembre 2008 (29093 *spams* et 8499 *hams*), période juste avant celle couverte par les flots de test. Ce classificateur a été ensuite utilisé pour classer chacun des flots. L'observation de l'évolution des résultats de classement, sans mise à jour du classificateur, permet d'observer l'évolution temporelle des caractéristiques des flots. Voir schéma dans la Figure 10.3.

[2]Les pièges à spam sont aussi connus par *pots de miel*, *honeypot* ou *spam trap*

Les ensembles *JM-H* et *JM-S* permettent de comparer les deux classes (ham et spam), tandis que l'observation simultanée des trois ensembles de spam permet de comparer trois flots de de la même classe reçus simultanément à des endroits différents et, à priori, sans aucun lien entre eux. Idéalement il serait nécessaire d'étudier et de comparer plusieurs flots de messages légitimes, mais l'acquisition de ce type de message sur une période aussi longue est irréaliste.

Ce classificateur peut être vu comme un "instrument de mesure" étalonné pour classer spécifiquement les ensembles *JM-H* et *JM-S*, puisque la distribution des messages dans ces flots est censée être identique à celle utilisée pour l'apprentissage du classificateur. L'utilisation de cet "instrument" pour classer les deux autres ensembles est juste un moyen d'étudier leur dérive.

Corpus	Classe	Messages	Classés Ham	Classés Spam	Taux d'erreur
JM-H	Ham	86673	84419	2254	2.601 %
JM-S	Spam	379374	837	378537	0.221 %
SB-S	Spam	487473	274	487199	0.056 %
BG-S	Spam	1222270	7972	1214298	0.652 %

TAB. 10.1: Ensembles de messages utilisés et résultats globaux de classement en boucle ouverte, avec un seuil de classement trivial.

Le classificateur utilisé est le classificateur discriminant linéaire décrit dans le Chapitre 9.

Le classement s'effectue selon la règle de décision triviale : on attribue le message à la classe *spam* si le score est positif et à la classe *ham* si négatif. La Table 10.1 présente les résultats globaux de classement utilisant cette règle de décision. Ce classement binaire n'est qu'indicatif et ne sert qu'à vérifier si les résultats sont réalistes.

Le score de chaque message a été enregistré et les résultats ont été agrégés par jour :
- le nombre de messages ;
- la moyenne journalière du score ;
- l'écart-type des scores de la journée ;
- l'entropie des scores de la journée.

Le nombre de messages par jour est un indicateur de l'intensité du flot.

Grâce à la *"fonction de transfert du classificateur"* Le score d'un message est la projection, sur une seule dimension, de la représentation vectorielle des messages. Dans cet expérimentation, on peut le voir comme un indicateur qui agrège, en une seule valeur numérique (un scalaire), l'ensemble des caractéristiques d'un message : le fond (le contenu effectif) et la forme (comment le message a été créé, mis en forme et distribué). Il s'agit d'une représentation avec perte : l'attribution d'un score à un message n'est pas une fonction injective puisque la même valeur de score peut être associée à des messages différents. En revanche, des scores différents correspondent toujours à des messages différents.

L'écart-type et l'entropie sont tous les deux, avec des points de vue différents, des indicateurs de la diversité d'un ensemble de messages : l'écart-type évalue la dispersion des scores autour de la moyenne, tandis que l'entropie évalue l'homogénéité de la distribution des scores. Aussi, il y a une certaine redondance entre l'écart-type et l'entropie - nous ne présentons les deux que lorsque cela devient utile.

L'entropie du score, a été évaluée à partir du histogramme du score comptabilisé sur la plage de valeurs découpée en 128 intervalles (*bins*), par l'expression :

$$\hat{H} = \sum_{i=0}^{127} \frac{N_i}{N} \, log_2 \left(\frac{1}{N_i/N} \right) \tag{10.1}$$

où N_i est le nombre de messages dont le score se trouve dans le *bin* i et N et le nombre total de messages. Dans l'absolu, cette méthode d'estimation de l'entropie n'a pas de sens, mais elle

nous est utile pour détecter, qualitativement, des évolutions ou des événements périodiques dans le flot de messages.

Un raisonnement simpliste justifie l'étude conjointe des séries de moyenne et écart-type du score. On peut imaginer l'ensemble de messages comme étant un nuage centré sur la moyenne du score et dont la dispersion est définie par l'écart-type. Si l'on suppose que la répartition des scores suit une distribution normale, alors 95% des messages seraient concentrés dans un voisinage de rayon 1.96 fois l'écart-type. Si le valeur du seuil de classement se trouve à l'intérieur de ce voisinage, alors l'erreur de classement est supérieure à 2.5%.

Nous utilisons des outils variés pour l'analyse de ces séries : le demi-variogramme pour détecter les dépendances temporelles et aussi la ressemblance ou différence entre deux flots de messages ; le périodogramme permettant de mettre en valeur des phénomènes périodiques et l'ajustement d'un modèle linéaire *SARIMA* pour identifier quantitativement les paramètres des séries temporelles des indicateurs des flots de messages.

10.4.2 Les données brutes

Les séries temporelles brutes des indicateurs sont présentées graphiquement dans les Figures 10.4 (nombre de messages par jour, en échelle logarithmique) et 10.6 (moyenne, écart-type et entropie du score).

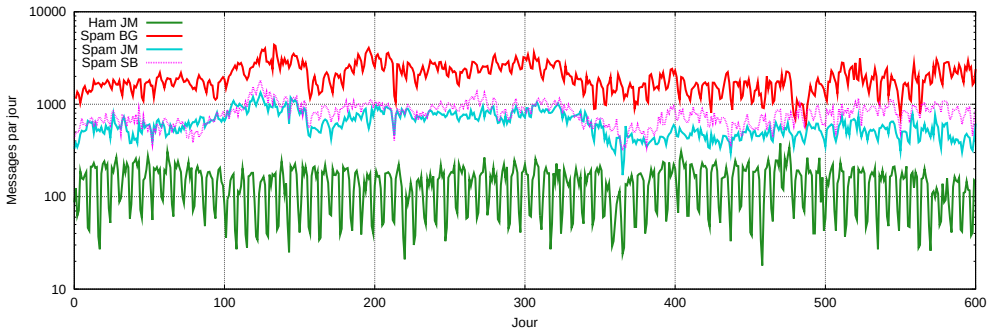
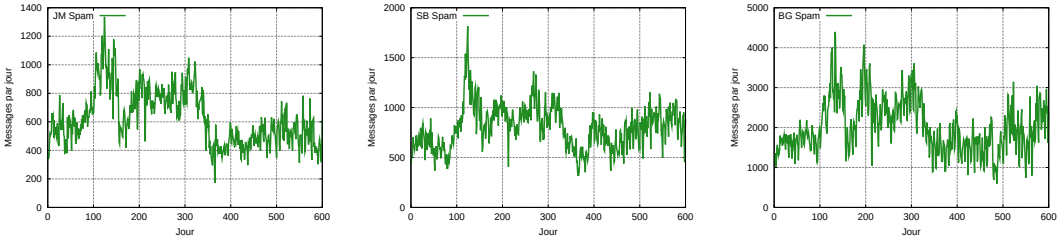


FIG. 10.4: Messages par jour dans des corpus JM-H, JM-S, SB-S et BG-S.

Le nombre de messages par jour : Dans la Section 10.2, nous avons mentionné l'évolution temporelle du taux d'arrivée de messages dans la boîte aux lettres de l'auteur (JM-H et JM-S) sur une période encore plus longue (32 mois) et constaté des périodicités régulières à 'intérieur de la journée et de la semaine. Dans l'expérimentation en cours, les variations à l'intérieur de la journée sont masquées par la taille de l'intervalle d'analyse.



(a) Messages par jour JM-S (b) Messages par jour SB-S (c) Messages par jour BG-S

FIG. 10.5: Les taux de messages des corpus de spam, lorsque représentés dans des échelles comparables (Figures 10.5a, 10.5b et 10.5c) suggèrent une forte ressemblance entre ces trois ensembles.

Le flot de messages légitimes, *hams*, (Figure 10.4) est assez régulier et marqué uniquement par la variation hebdomadaire que l'on explique par la baisse d'activité professionnelle pendant le weekend. Les flots de la classe *spam* (Figure 10.4) sont, visiblement, indépendants de celui de la classe *ham*. Ils contiennent, certes, une composante périodique hebdomadaire, plus visible dans le flot BG-S, mais avec intensité plus faible : la composante dominante est plutôt erratique et résulte probablement des changements du niveau d'activité des *spammeurs*.

La ressemblance dans le nombre de messages par jour des flots de spam - *JM-S*, *SB-S* et *BG-S* - apparait visuellement dans la Figure 10.5 lorsqu'ils sont représentés dans des échelles comparables. Une façon d'estimer cette ressemblance est de considérer ces séries comme des vecteurs et d'évaluer, pour chaque couple, le cosinus de l'angle entre les vecteurs ainsi que la distance entre les vecteurs normalisés :

$$\begin{aligned} cosinus(\vec{s_1}, \vec{s_2}) &= \frac{< \vec{s_1}, \vec{s_2} >}{|\vec{s_1}| \cdot |\vec{s_2}|} \\ distance(\vec{s_1}, \vec{s_2}) &= \left| \frac{\vec{s_1}}{|\vec{s_1}|} - \frac{\vec{s_2}}{|\vec{s_2}|} \right| \end{aligned} \tag{10.2}$$

Si la valeur du cosinus est unitaire ou si cette distance normalisée est nulle (ce sont des situations équivalentes), alors les vecteurs sont colinéaires et liés par un facteur d'échelle.

L'évaluation de ces fonctions pour chacune des pairs de séries de spams, présenté dans la Table 10.2, suggère que ces trois séries se ressemblent plutôt qu'elles différent.

	JM-S	*SB-S*	*BG-S*
JM-S	1.0000 / 0.0000	0.9778 / 0.2105	0.9759 / 0.2194
SB-S	0.9778 / 0.2105	1.0000 / 0.0000	0.9745 / 0.2257
BG-S	0.9759 / 0.2194	0.9745 / 0.2257	1.0000 / 0.0000

TAB. 10.2: Le cosinus et distance, définis par 10.2, entre les séries du nombre de messages par jour des flots *JM-S*, *SB-S* et *BG-S* montrent la ressemblance temporelle entre ces séries.

Les indicateurs de classement : La comparaison visuelle des séries des indicateurs de classement (voir Figure 10.6) suggère que les flots de *spams* sont similaires entre eux, mais différents des *hams*.

Pour l'ensemble des *hams*, les deux moments (moyenne et écart-type du score) ont un comportement assez régulier : des valeurs concentrées autour d'une moyenne, présentant une faible dérive linéaire et une dispersion plus ou moins constante. Ce qui n'est pas le cas pour les trois ensembles de *spam* qui, en plus d'une tendance linéaire apparente, présentent des évolutions irrégulières mais visuellement similaires.

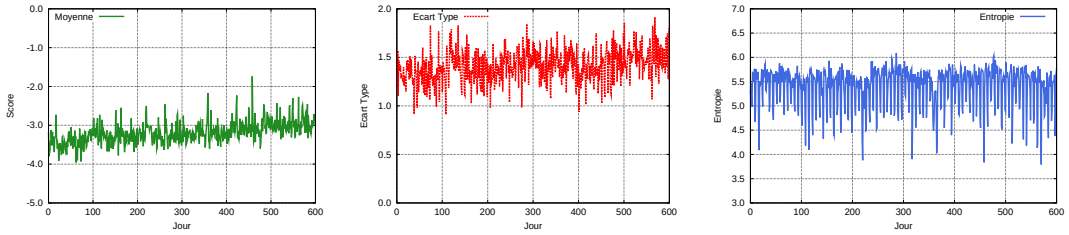
La dérive linéaire de la moyenne du score s'oriente vers la valeur du seuil de classement.

On remarque, visuellement, des ressemblances entre les séries d'écart-type du score des *spams* : un creux juste après le jour 100, puis un pic juste après le jour 300 et la variation moins régulière dans *spams* que dans les *hams*.

10.4.3 La tendance à long terme (dérive linéaire) des séries

Dans la subsection précédente nous avons observé visuellement que les séries des indicateurs présentent une dérive temporelle, en apparence, linéaire. Cette dérive peut être évaluée grâce à une régression linéaire, si l'on décompose chaque série en :

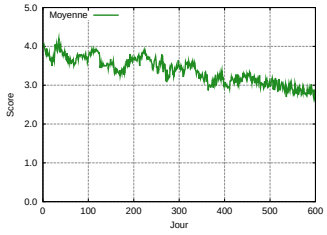
$$X(t) = (\alpha + \beta.t) + R(t) \tag{10.3}$$



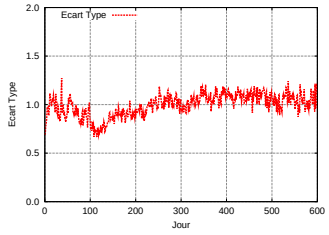
(a) Moyenne du score - JM-H

(b) Écart-type du score - JM-H

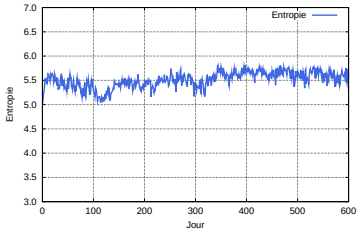
(c) Entropie du score - JM-H



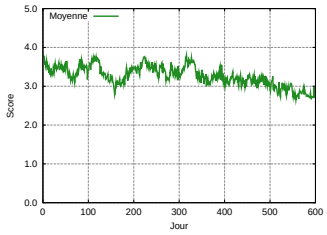
(d) Moyenne du score - JM-S



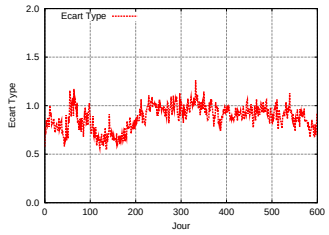
(e) Écart-type du score - JM-S



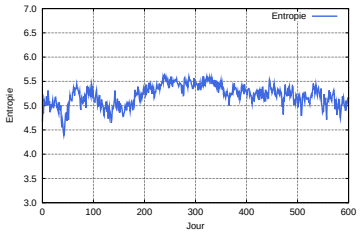
(f) Entropie du score - JM-S



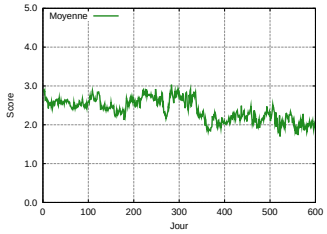
(g) Moyenne du score - SB-S



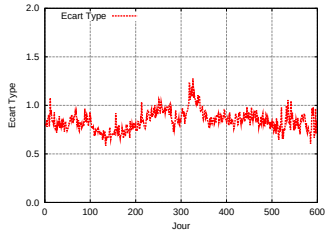
(h) Écart-type du score - SB-S



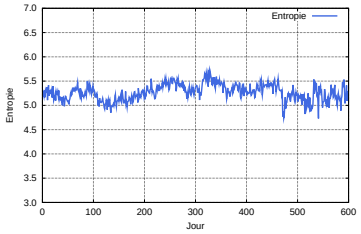
(i) Entropie du score - SB-S



(j) Moyenne du score - BG-S



(k) Écart-type du score - BG-S



(l) Entropie du score - BG-S

FIG. 10.6: Séries temporelles journalières construites à partir de la moyenne et écart-type du score de classement des flots *JM-H*, *JM-S*, *SB-S* et *BG-S*

où $(\alpha + \beta.t)$ est une tendance déterministe linéaire et $R(t)$ est un processus stochastique de moyenne *nulle*. Les valeurs de α et β, ont été estimées pour chaque série à l'aide de la bibliothèque de régression linéaire *glm* du logiciel *R* et présentées dans la Table 10.3.

	Indicateur	$\hat{\alpha}$		$\hat{\beta}$	
JM-H	Score Moyen	$-3.442\ (\pm 3.374\ 10^{-2})$	***	$8.425\ 10^{-4}\ (\pm 1.067\ 10^{-4})$	***
	Écart-Type	$1.323\ (\pm 2.591\ 10^{-2})$	***	$2.862\ 10^{-4}\ (\pm 7.405\ 10^{-5})$	***
	Entropie	$5.670\ (\pm 6.325\ 10^{-2})$	***	$-2.310\ 10^{-4}\ (\pm 1.807\ 10^{-4})$	*
JM-S	Score Moyen	$3.897\ (\pm 2.425\ 10^{-2})$	***	$-1.773\ 10^{-3}\ (\pm 6.929\ 10^{-5})$	***
	Écart-Type	$0.8782\ (\pm 1.459\ 10^{-2})$	***	$3.802\ 10^{-4}\ (\pm 4.169\ 10^{-5})$	***
	Entropie	$6.123\ (\pm 3.220\ 10^{-2})$	***	$2.816\ 10^{-5}\ (\pm 9.198\ 10^{-5})$	
SB-S	Score Moyen	$3.579\ (\pm 3.028\ 10^{-2})$	***	$-1.072\ 10^{-3}\ (\pm 8.649\ 10^{-5})$	***
	Écart-Type	$0.8113\ (\pm 1.896\ 10^{-2})$	***	$2.165\ 10^{-4}\ (\pm 5.417\ 10^{-5})$	***
	Entropie	$5.937\ (\pm 6.968\ 10^{-2})$	***	$3.110\ 10^{-4}\ (\pm 1.991\ 10^{-4})$	**
BG-S	Score Moyen	$2.729\ (\pm 3.352\ 10^{-2})$	***	$-1.166\ 10^{-3}\ (\pm 9.375\ 10^{-5})$	***
	Écart-Type	$0.8180\ (\pm 1.568\ 10^{-2})$	***	$7.700\ 10^{-5}\ (\pm 4.469\ 10^{-5})$	***
	Entropie	$5.978\ (\pm 4.047\ 10^{-2})$	***	$-8.878\ 10^{-5}\ (\pm 1.153\ 10^{-4})$	

Tab. 10.3: Estimation de la tendance déterministe linéaire des indicateurs obtenue par ajustement d'un modèle linéaire $(\hat{\alpha} + \hat{\beta}.t)$. Ces paramètres, avec des intervalles de confiance à 95% ont été obtenus à l'aide du logiciel *R*. Les intervalles de confiance montrent que, à l'exception des pentes de l'entropie, ces valeurs sont statistiquement significatives (l'indication *** est attribuée par *R* pour p-value $< 10^{-3}$, et en fait les valeurs de p-value sont bien inférieures à cela).

La dérive de la moyenne des scores est toujours inférieure à 1/1000 de la valeur initiale : positive pour les *hams* et négative pour les *spams*. On peut interpréter cette dérive si l'on imagine les deux classes de messages comme des nuages dont les centres convergent vers un point commun. Ce point commun étant le seuil de classement, cela confirme l'intuition que, sans mise à jour du classificateur, le taux d'erreur ne peut qu'augmenter avec le temps.

Néanmoins, cette faible dérive indique que, au moins dans le cas de ce classificateur et de ces flots, la mise à jour en permanence du classificateur avec des exemples très récents, ne constitue pas une obligation à respecter absolument. Si cette remarque ne peut pas être généralisée, on peut néanmoins penser qu'il peut avoir des classificateurs construits astucieusement de façon à ce que cette contrainte ne soit pas absolue.

L'écart-type du score est plus important pour l'ensemble de *hams* (autour de 1.3) que pour les ensembles de *spam* (autour de 0.8) ce qui suggère que la diversité des messages, perçue par le classificateur, est plus importante dans la classe des *hams*. Cela confirme l'intuition que l'on peut avoir puisque les spams sont, la plus part du temps, répétitifs.

La dérive de l'écart-type est positive dans tous les cas, mais encore plus faible que celle de la moyenne des scores. Néanmoins, étant toujours positive, cela suggère que la dispersion du score moyen augmente. Cette expérimentation ne permet pas de déduire directement si cela résulte d'une augmentation effective de la diversité des messages ou d'une dépendance existante entre la moyenne et l'écart-type des scores.

Les résultats de régression linéaire des indicateurs des flots de *spam* présentent des valeurs proches aussi bien pour la moyenne que pour l'écart-type, à l'exception de la moyenne du score du flot BG-S.

10.4.4 Les demi-variogrammes des séries

Dans l'étude de séries temporelles, les corrélogrammes sont utilisés plus souvent que les variogrammes. Ce dernier est plus courant dans l'étude de données spatiales, en particulier la geostatistique, domaine où il a été proposé. Si bien que les deux outils peuvent être utilisés pour le traitement de séries temporelles, Cressie [76, p. 70] énumère une série de critères qui lui font privilégier le variogramme plutôt que le corrélogramme. Parmi ces critères, le plus intéressant, dans notre contexte, est l'indépendance d'une estimation de la moyenne de la série. Cela fait que le variogramme est encore défini lorsque la moyenne n'est pas constante. Bien sur, dans ce cas la série n'est pas stationnaire et son interprétation doit être faite avec précaution. Pour plus de détails sur la définition des variogrammes et de leur utilisation, consulter le résumé dans l'Annexe B, Chiles et Delphiner [49] ou encore Cressie [76].

Nous estimons le *demi-variogramme* par :

$$\hat{\gamma}(h) = \frac{1}{2(N-h)} \sum_{x=0}^{N-1-h} \left(Z(x) - Z(x+h)\right)^2 \tag{10.4}$$

Dans la pratique, lorsque la longueur de la série est finie, l'estimation des variogrammes se limite à des valeurs de pas inférieures à 1/3 ou 1/2 de la longueur de la série : le nombre de points pris en compte décroit avec la valeur du pas (dont la signification statistique de l'estimation en dépend). Néanmoins, l'estimation de ces variogrammes est étendue jusqu'à 2/3 (soit 400 points) - l'information perd en signification statistique, mais on peut encore l'utiliser dans un but qualitatif pour comparer des variogrammes de séries différentes.

Le *demi-variogramme* normalisé par la *variance* nous permet d'observer des séries différentes dans une même échelle, puisque la valeur asymptotique du demi-variogramme est égale à la variance du processus, si le processus est stationnaire de deuxième ordre.

Les demi-variogrammes ont été regroupés dans la Figure 10.7 pour les indicateurs des séries de la classe spam (JM-S, SB-S et BG-S) et dans la Figure 10.8 pour ceux des séries de l'auteur (JM-H et JM-S). À l'exception des indicateurs d'entropie, ces variogrammes ont été estimés après élimination de la dérive déterministe linéaire (cf Table 10.3).

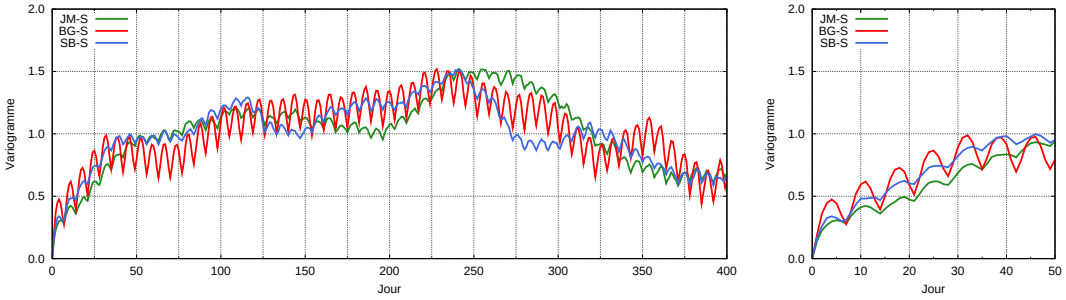
On remarque, tout d'abord, une ressemblance entre les séries des indicateurs des flots de *spams*, grâce à leurs variogrammes (Figure 10.7). Cette ressemblance est confirmée par les demi-variogrammes croisés (Figure 10.9), qui non seulement se ressemblent mais ressemblent aussi les variogrammes simples. Le variogramme du nombre de messages par jour présente une composante périodique hebdomadaire qui semble ne pas impacter les variogrammes des indicateurs de classement (moyenne et écart-type du score).

Alors que l'on remarque une ressemblance entre les séries des flots de *spam*, grâce à leurs indicateurs, on remarque que ces séries sont bien différentes de celles du flot de *hams*. Aussi, la composante périodique du nombre de messages par jour est bien plus importante dans le flot de *hams* que dans le flot de *spams*, avec un impact non négligeable dans les indicateurs de classement. Cet aspect est confirmé et commenté dans les sections qui suivent.

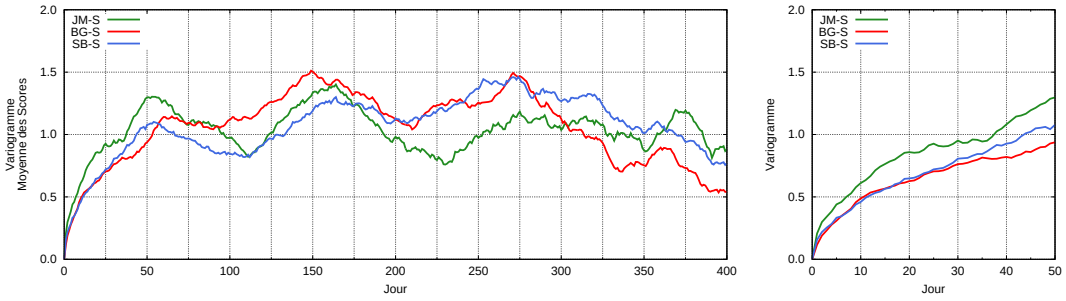
10.4.5 Caractérisation par ajustement d'un modèle dynamique linéaire

Une méthode souvent utilisée pour caractériser une série temporelle est l'ajustement d'un modèle dynamique sous-jacent, linéaire ou pas, susceptible de générer la série en étude à partir d'un bruit blanc : un processus purement aléatoire, stationnaire et non corrélé [48] [124] [38]. Les modèles linéaires courants sont les modèles AR, MA, ARMA, ARIMA et SARIMA (voir définition de ces modèles dans l'Annexe B).

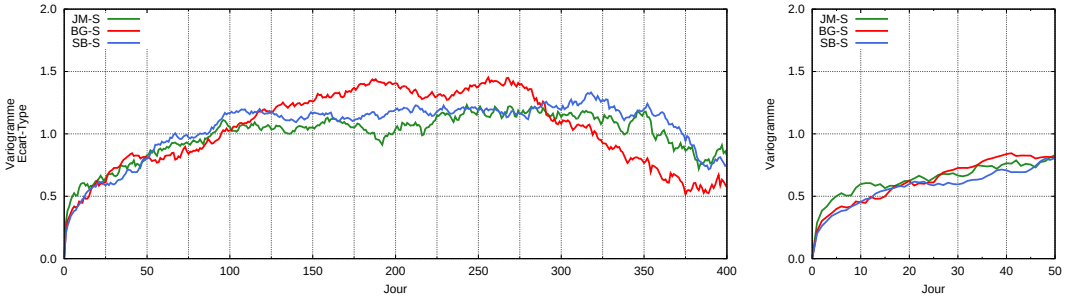
L'identification du modèle associé à chaque série a été faite à l'aide du logiciel statistique *R* [205]. La fonction `arima` de *R* ajuste, pour un 6-tuple, définissant l'ordre du modèle, les quatre polynômes du modèle dynamique *SARIMA* correspondant - voir Equation (B.18) - les plus vraisemblables pour cet ordre et retourne la valeur de la vraisemblance de l'ajustement. Un balayage exhaustif de l'ensemble des 6-tuples possibles permet de choisir le modèle ayant



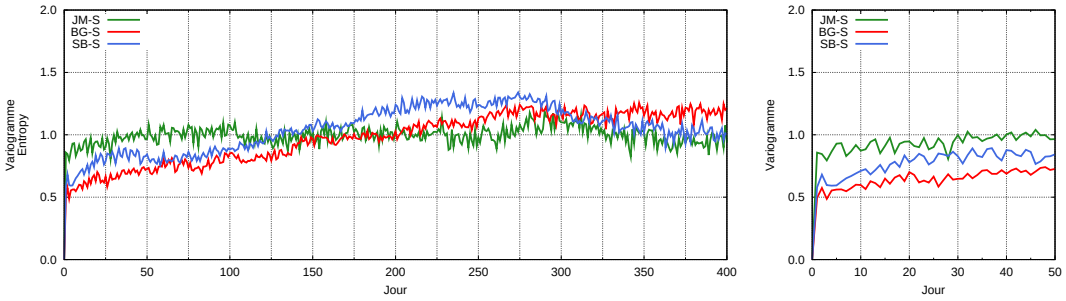
(a) Messages par jour



(b) Moyenne des scores

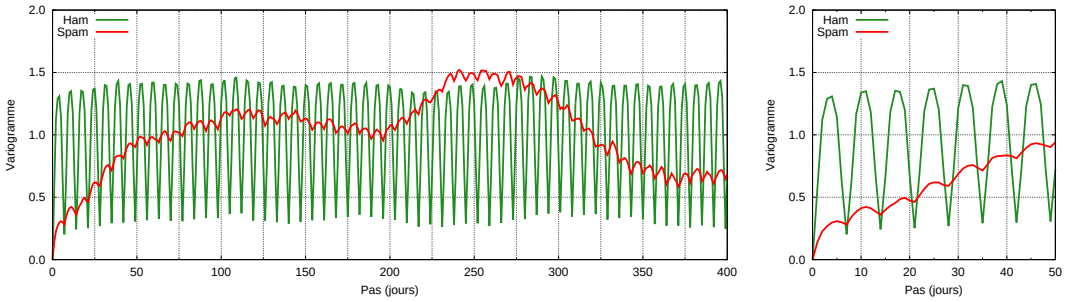


(c) Écart type des scores

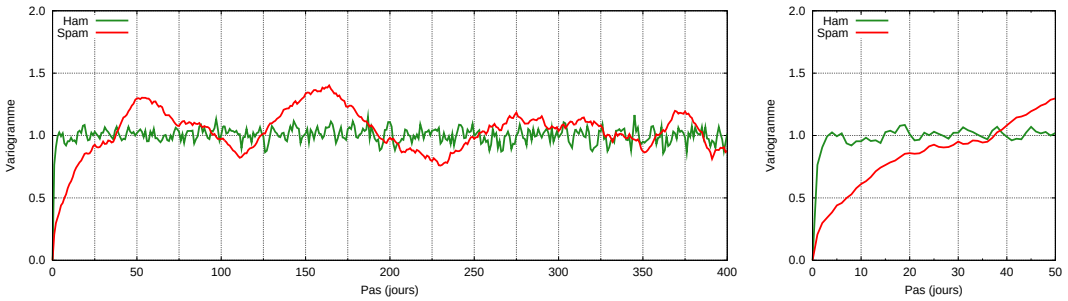


(d) Entropie des scores

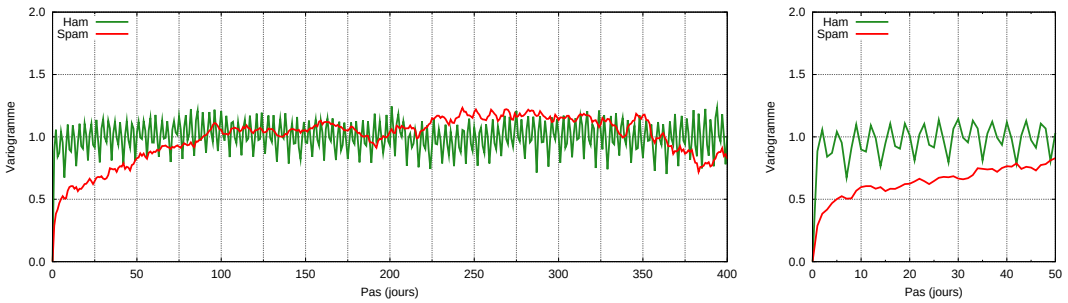
FIG. 10.7: Variogrammes simples des séries des indicateurs des flots *JM-S SB-S* et *BG-S* après élimination de la tendance déterministe linéaire.



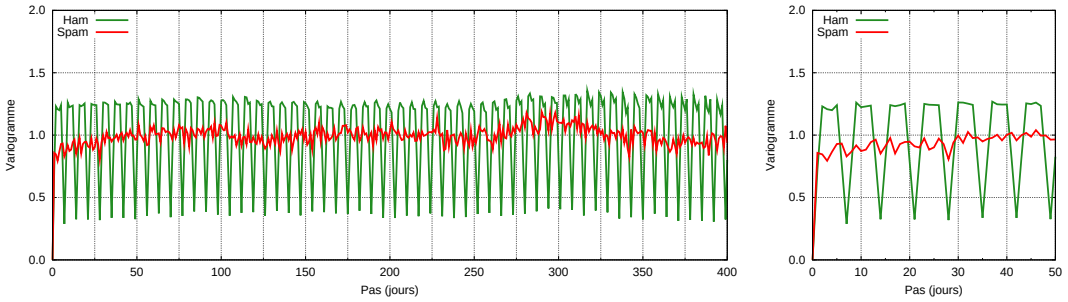
(a) Nombre de messages par jour



(b) Moyenne des scores



(c) Écart-type des scores

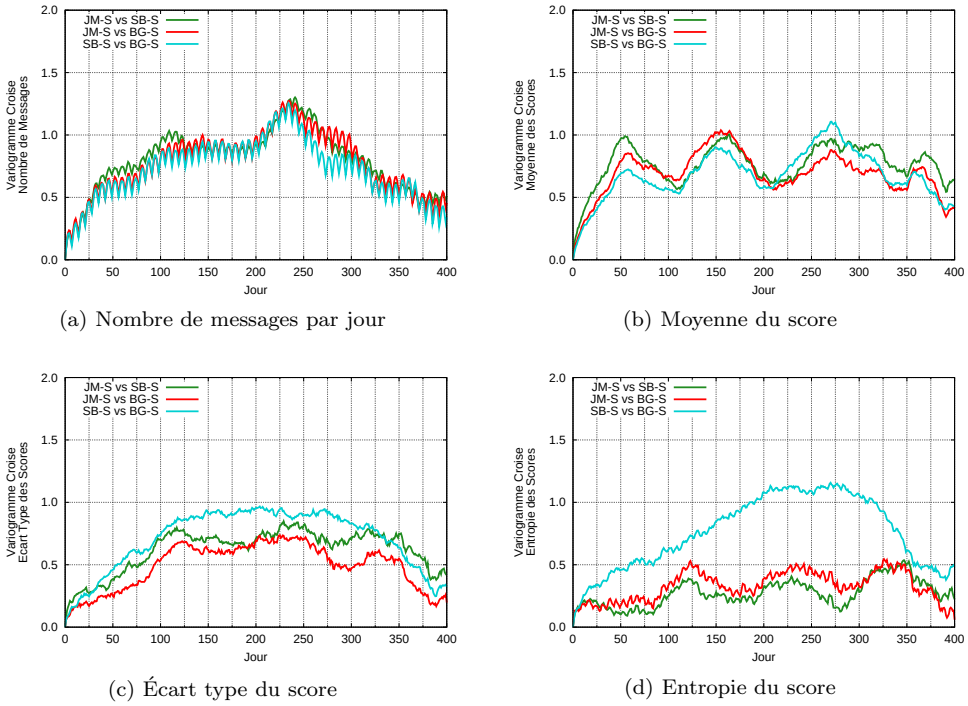


(d) Entropie des scores

FIG. 10.8: Demi-variogrammes des séries des indicateurs (messages par jour, moyenne, écart-type et entropie des score) des les flots JM-H et JM-S, après élimination de la tendance déterministe linéaire.

	Modèle	Score Moyen Partie non saisonnière	Partie Saisonnière
JM-H	ARIMA(1,0,0)	$\phi = (0.2328)$ $\phi_{se} = (0.0395)$	
JM-S	ARIMA(1,0,1)	$\phi = (0.9155)\ \theta = (-0.3576)$ $\phi_{se} = (0.0213)\ \theta_{se} = (0.0522)$	
SB-S	SARIMA(1,0,1)(1,0,1)$_7$	$\phi = (0.9433)\ \theta = (-0.3699)$ $\phi_{se} = (0.0170)\ \theta_{se} = (0.0468)$	$\Phi = (0.9840)\ \Theta = (-0.9528)$ $\Phi_{se} = (0.0178)\ \Theta_{se} = (0.0317)$
BG-S	ARIMA(1,0,1)	$\phi = (0.9320)\ \theta = (-0.2329)$ $\phi_{se} = (0.0171)\ \theta_{se} = (0.0493)$	
	Modèle	**Erreur-Type** **Partie non saisonnière**	**Partie Saisonnière**
JM-H	SARIMA(0,0,0)(2,0,0)$_7$		$\Phi = (0.2858, 0.1532)$ $\Phi_{se} = (0.0403, 0.0406)$
JM-S	ARIMA(2,0,1)	$\phi = (1.2535, -0.2813)\ \theta = (-0.7606)$ $\phi_{se} = (0.0944, 0.0831)\ \theta_{se} = (0.0775)$	
SB-S	ARIMA(1,0,1)	$\phi = (0.9468)\ \theta = (-0.4779)$ $\phi_{se} = (0.0160)\ \theta_{se} = (0.0480)$	
BG-S	ARIMA(2,0,1)	$\phi = (1.3518, -0.3699)\ \theta = (-0.7802)$ $\phi_{se} = (0.0720, 0.0669)\ \theta_{se} = (0.0527)$	

TAB. 10.4: Modèles ajustés pour les séries de moyenne et écart-type du score des flots. La ligne inférieure de chaque case correspond à l'erreur type de chaque coefficient et permet d'évaluer l'intervalle de confiance de la valeur des coefficients ajustés.



(a) Nombre de messages par jour

(b) Moyenne du score

(c) Écart type du score

(d) Entropie du score

FIG. 10.9: Les demi-variogrammes croisés des séries des indicateurs des flots de spam *JM-S SB-S* et *BG-S* après élimination de la tendance déterministe linéaire renforcent l'hypothèse de l'existence d'un lien de dépendance, au moins partielle, entre ces flots.

la valeur de *BIC* le plus faible. Finalement, pour valider ce choix, il faut encore vérifier que la série Z_t résiduelle est bien stationnaire et non corrélée. Cette étape a été faite visuellement : la série résiduelle brute permet de constater que la moyenne et variance sont approximativement constantes et la autocorrélation (fonction `acf`) doit être "négligeable" pour toute valeur de pas non nulle. On entend par "négligeables" les valeurs d'autocorrélation dont la valeur absolue est inférieure à $1.96/\sqrt{N}$, intervalle de confiance à 95 % pour une valeur nulle, où N est le nombre de termes de la série [38] [74][3].

Cette démarche est similaire à celle proposée par Cowpertwait [74, p. 144], à l'exception que, comme méthode d'ajustement nous avons utilisé la vraisemblance maximale (*ML*) au lieu de la somme quadratique conditionnelle (*CSS*) et que comme critère de sélection du modèle nous avons utilisé le *BIC* au lieu de *AIC*[4].

Les modèles trouvés par cette démarche sont présentés dans la Table 10.4[5], dans le format de R avec les coefficients des différents polynômes. Néanmoins, une interprétation plus intéressante utilise plutôt les racines de ces polynômes, ce sont ces valeurs qui définissent le comportement dynamique des séries[6]. Et c'est ce que nous utilisons pour interpréter les modèles des séries des flots de spam.

Les modèles ajustés pour le flot de *hams* ne ressemblent pas à ceux des flots de *spam*, ce qui suggère, de toute évidence, que les flots des deux classes ont des dynamiques différentes.

Dans les séries de la moyenne du score des flots de *spam*, la partie ARMA des modèles

[3] Ceci correspond au quantile 0.975 d'une distribution normale standard

[4] *BIC* - Bayesian Information Criteria et *AIC* - Akaike Information Criteria - (voir définition de ces critères dans l'Annexe A)

[5] Le logiciel R utilise une convention différente de celle de l'Équation B.18 en ce qui concerne le signe des coefficients des polynômes de la partie MA - voir fonction `arima` dans [205, p. 1091].

[6] Autrement dit, les racines de ces polynômes correspondent aux *zeros* et aux *pôles* de la transformée en $\mathcal{Z}$ d'une fonction de transfert

	Modèle	AR		MA	BIC
JM-S	ARMA(2,1)	0.9607	0.2928	0.7606	-1632.08
SB-S	ARMA(1,1)	0.9468	-	0.4779	-1512.75
BG-S	ARMA(2,1)	0.9708	0.3810	0.7802	-1752.50
SB-S*	ARMA(2,1)	0.9616	0.1772	0.6430	-1509.47

TAB. 10.5: Modèles ajustés pour les séries d'écart-type des scores des flots de spam. On reprend ici les résultats de la Table 10.4, mais les résultats présentés sont sous la forme des racines des polynômes au lieu de leurs coefficients. La ligne *SB-S** correspond au modèle *ARMA(2,1)* ajusté au flot *SB-S*, mais qui n'a pas été retenu, vu que son *BIC* est supérieur à celui du modèle *ARMA(1,1)*.

est fortement similaire pour les trois flots. Le modèle du flot SB-S est le seul qui présente une composante saisonnière. Si l'on observe les périodogrammes de ces séries on remarque que, effectivement, toutes les trois présentent une petite composante saisonnière et que le flot *SB-S* est celui qui possède la composante saisonnière la plus importante.

Pour les flots de *spam*, les modèles ajustés aux séries de l'écart-type sont tous du type *ARMA*. Néanmoins la partie *AR* du flot *SB-S* est d'un ordre inférieur et les coefficients sont différents. Si l'on observe plutôt les racines des polynômes (voir Table 10.5), on remarque que, même si l'ordre est différent, la valeur de la racine la plus proche de la frontière de stabilité (racine unitaire) de la partie *AR* est presque identique pour les trois flots : 0.9607, 0.9468 et 0.9708.

Encore dans cette table, la ligne *SB-S** correspond à un modèle *ARMA(2,1)* ajusté au flot *SB-S*, identique à celui ajusté pour les deux autres flots. On remarque que les valeurs du critère *BIC* des deux modèles est assez proche (-1512 contre -1509), les valeurs du log de la vraisemblance sont quasiment identiques (769.2 contre 770.7) et c'est juste la complexité du modèle qui a été déterminante dans le choix du modèle.

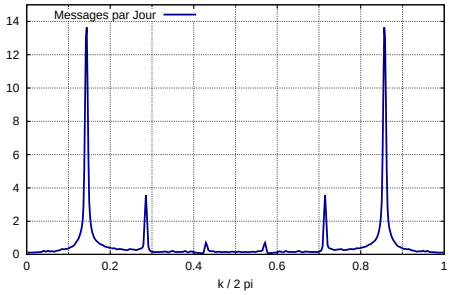
Une dernière remarque concerne l'ordre des modèles ajustés des séries de moyenne et écart-type du score. Aucun de ces modèles ne contient une racine unitaire dans le polynôme caractérisant la partie auto-régressive de la série, et cela même si l'on tient compte de l'erreur estimée d'ajustement. Autrement dit, aucune des séries n'est devenue stationnaire par dérivation de la série d'origine. Par contre, tous les séries de la classe *spam*, et seulement de cette classe, contiennent une racine assez proche de l'unité : environ 0.93 pour les séries de la moyenne du score et 0.96 pour les séries de l'écart-type du score. Cela suggère que la prévisibilité, par rapport à la veille, des indicateurs de la classe *spam* est meilleure que celle de la classe *ham*.

10.4.6 Analyse Spectrale

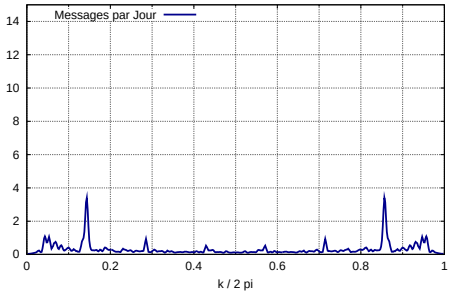
La transformation de Fourier du correlogramme[7] d'une série permet de l'étudier dans le domaine des fréquences et d'identifier les événements périodiques de la série. Le périodogramme est un estimateur de la transformation de Fourier du corrélogramme.

Pour l'estimation des périodogrammes, les séries ont été traitées pour éliminer la composante linéaire, estimée auparavant, ainsi que la composante irrégulière lente : d'une part nous souhaitons étudier les composantes périodiques dont la périodicité est de l'ordre de la semaine et d'autre part ces composantes contribuent à rendre non stationnaires les séries (un périodogramme n'a de sens que pour des signaux stationnaires). La composante irrégulière lente a été supprimée à l'aide d'un simple filtre passe-hautes : à chaque point de la série, on soustrait la valeur moyenne évaluée sur les 32 points voisins). Pour réduire la variance de l'estimateur du périodogramme, un

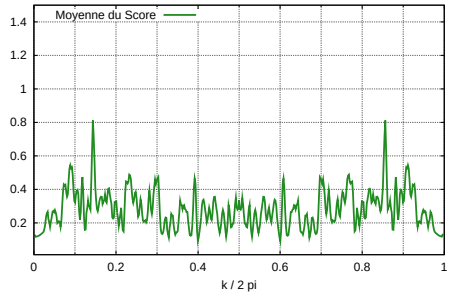
[7]La transformation de Fourier de l'autocorrelation d'un signal aléatoire est connue, en traitement de signal ou en télécommunications, par la désignation *Densité Spectrale de Puissance*



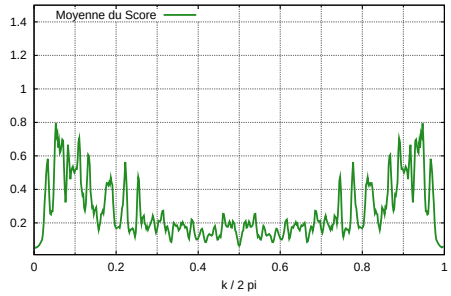
(a) JM-H - Messages par jour



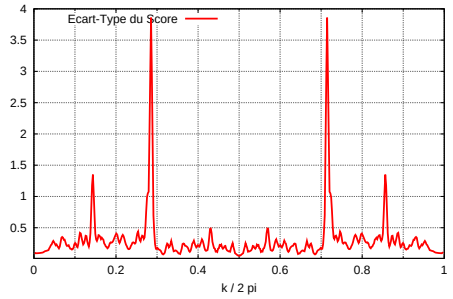
(b) JM-S - Messages par jour



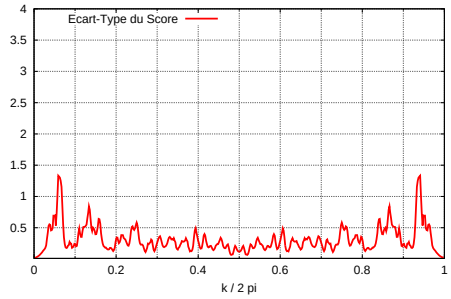
(c) JM-H - Moyenne des scores



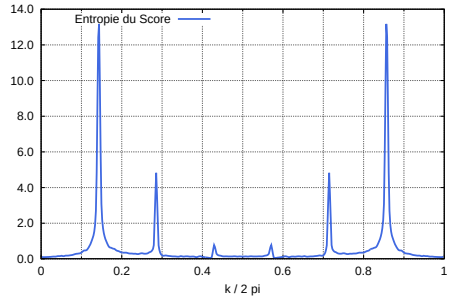
(d) JM-S - Moyenne des scores



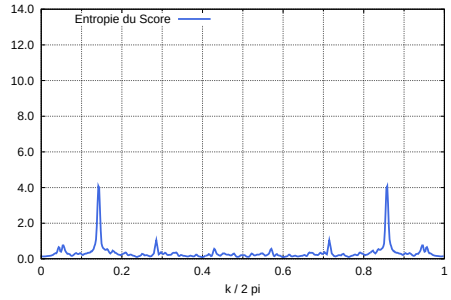
(e) JM-H - Ecart type des scores



(f) JM-S - Ecart type des scores

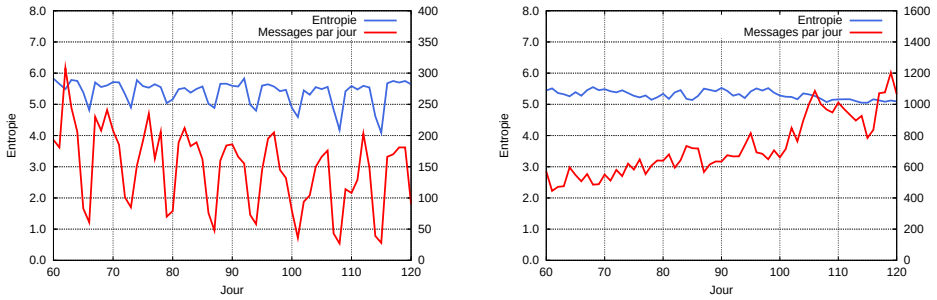


(g) JM-H - Entropie des Scores



(h) JM-S - Entropie des Scores

FIG. 10.10: Périodogrammes (échelle logarithmique), des séries des indicateurs des flots *JM-H* et *JM-S*, après élimination de la tendance déterministe linéaire.



(a) JM-H (b) JM-S

FIG. 10.11: Superposition, sur deux mois (troisième et quatrième), des séries de l'entropie et du nombre de messages par jour des flots JM-H et JM-S. Dans le flot JM-H, la coïncidence des creux dans les deux indicateurs montre que la baisse d'entropie (et de la diversité des messages) correspond bien aux fins de semaine. Dans le flot JM-S, l'entropie des scores n'est pas impactée par les variations du nombre de messages par jour.

lissage par un filtre triangulaire de Bartlett (fenêtre symétrique de taille deux points de chaque coté) a été appliqué et ajusté visuellement (voir l'Annexe B ou [48, p. 133]). Cette méthode est suffisante pour notre objectif.

Comme nous avons fait pour l'estimation des demi-variogrammes, les corrélogrammes sont aussi normalisés par rapport à la variance, ce qui permet la représentation de périodogrammes de séries différentes dans une échelle unique et la comparaison, y compris visuelle, de l'importance relative des composantes spectrales. Le corrélogramme normalisé est estimé par :

$$\rho(\hat{h}) = \frac{1}{(N-h).\hat{\sigma}^2} \sum_{x=1}^{N-h-1} [(Z(x) - \hat{\mu}) \cdot (Z(x+h) - \hat{\mu})] \tag{10.5}$$

Le périodogramme est estimé par :

$$\hat{S}(k) = \sum_{h=0}^{N-1} \hat{\rho}(h) \cdot e^{-i \frac{2\pi h}{N} k},\ h = 0, \ldots, N-1 \tag{10.6}$$

et lissé selon (avec $M = 2$) :

$$\widehat{S}'(k) = \sum_{i=-M}^{M} \frac{M+1-|i|}{(M+1)^2} \hat{S}(k+i)\ ,\quad t = 0, \ldots, N-1 \tag{10.7}$$

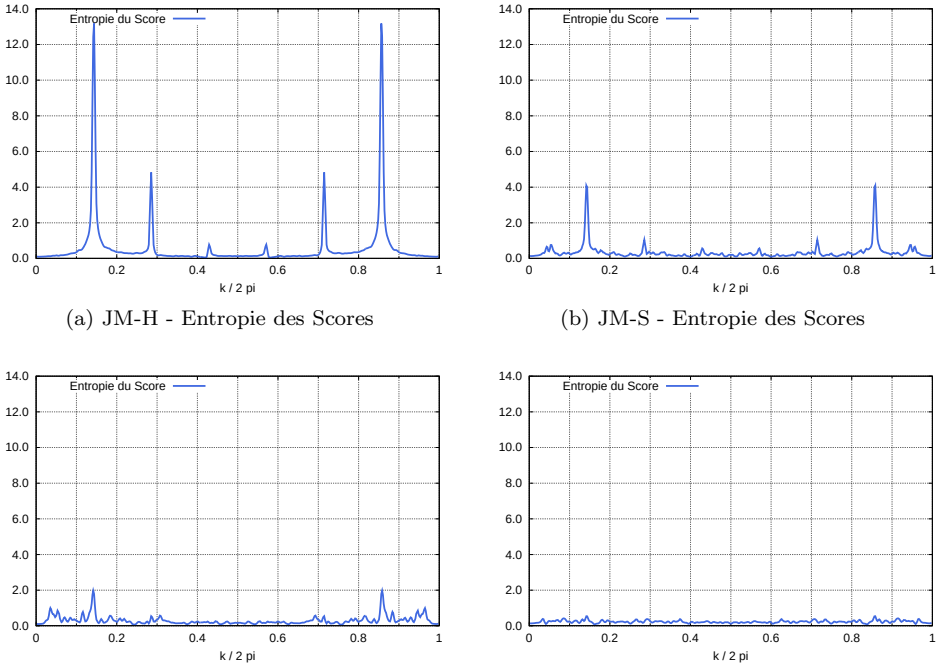
Les estimations des périodogrammes des indicateurs des flots de l'auteur (Figure 10.10)[8] confirment et mettent en valeur les différences les plus importantes entre les deux classes de messages.

Le caractère périodique du nombre de messages par jour avait déjà été remarqué auparavant, mais ces périodogrammes montrent clairement que la présence d'une composante périodique dans l'ensemble des indicateurs et plus forte dans le flot de *hams* que dans celui des *spams*.

Ce caractère périodique est indiqué par la raie placée à la coordonnée $2\,\pi/7$, soit environ 0.143 et les harmoniques multiples. La multiplicité des harmoniques, en particulier les harmoniques paires, s'explique par l'asymétrie du signal : un nombre de messages important pendant les cinq jours ouvrables de la semaine et une baisse pendant les deux jours du week-end.

La présence de cette périodicité sur les indicateurs de classement a une interprétation particulière. En effet, si la variation du nombre de messages par jour résultait uniquement d'un

[8] Les périodogrammes étant des fonctions complexes, il s'agit du module dans l'intervalle $[0, 2\pi]$.



(a) JM-H - Entropie des Scores

(b) JM-S - Entropie des Scores

(c) SB-S - Entropie des Scores

(d) BG-S - Entropie des Scores

FIG. 10.12: Périodogrammes des séries d'entropie des scores des flots *JM-H* et *JM-S*, *SB-S* et *BG-S*. Le périodogramme du flot de *hams* a une raie spectrale qui se détache clairement tandis que les flots de *spams* n'en ont pas.

échantillonnage plus ou moins important d'une même population, alors il ne devait pas avoir d'impact dans les indicateurs de classement, autres que les erreurs d'estimation. L'impact sur la moyenne, l'écart-type et l'entropie indique que les messages des jours ouvrables et du weekend ont pour origine des populations avec des distributions différentes.

Le flot de messages légitimes *JM-H* correspond à une boîte aux lettres professionnelle et suit l'activité de la personne associée. Il est constitué de deux catégories de messages : les *messages interactifs* résultant d'un échange entre le destinataire des messages et ses interlocuteurs et les *messages non interactifs* (p. ex., messages automatiques et listes de diffusion). Pendant les jours de la semaine et plus particulièrement pendant les heures de travail, le flot est constitué des deux catégories alors que pendant le weekend, le flot est constitué presque exclusivement des messages non interactifs.

La Figure 10.11a montre que la diminution de l'entropie et du nombre de messages légitimes dans le flot JM-H sont des événements synchrones et confirme que la diversité des messages est plus faible dans le weekend qu'en semaine.

En fait, malgré un ajustement satisfaisant de modèles linéaires (*AR*, *ARMA* et *SARIMA*) à chacune des séries temporelles prise individuellement, globalement il s'agit plutôt d'un modèle non linéaire à commutation de régime avec au moins deux régimes différents et dont le basculement est progressif : l'arrêt et le démarrage de l'activité professionnel de chacun des correspondants ne sont pas des événements synchrones).

Les périodogrammes de l'entropie du score des quatre flots (Figure 10.12) suggère que, effectivement, la composante périodique est relativement plus importante dans le flot de hams que dans le flot de spams. Par contre, dans les trois flots de spams, l'importance de la composante périodique n'est pas identique. Nous avons observé que dans les flots JM-S et SB-S, des flots, partie des spams résultent de messages publicitaires distribués par des entreprises de marketing

et non pas par des réseaux de *botnets*[9]. Cela explique que seulement une partie du spam reçu dans ces deux flots correspond à une activité cyclique, vraisemblablement d'importance plus faible que celle des hams.

Rappelons que l'unité d'agrégation des indicateurs étant la journée, nous n'avons pu identifier que les périodicités hebdomadaires. Nous avons vu dans la Section 10.2 que, pour raisons identiques, l'intensité du flot de messages varie dans la journée, avec des pics dans la journée et des creux dans la nuit, à cause du cycle d'activité. Une étude avec une granularité plus fine pourrait permettre d'identifier cela, mais la densité de messages disponible n'est pas suffisante : nous n'avons que quelques messages à l'heure.

10.4.7 Comparaison avec autres classificateurs

Les résultats présentés jusqu'ici ont été obtenus avec un classificateur basé sur un algorithme discriminant linéaire *SLDC*. Dans cette sub-section, nous comparons ces résultats avec d'autres variantes du même classificateur et aussi avec un classificateur bayésien naïf. L'algorithme de classement *SLDC* appartient à la famille des classificateurs discriminants tandis que le classificateur bayésien naïf à celle des classificateurs génératifs.

Les attributs utilisés par *SLDC* sont des 4-grams de niveau caractère extraits des en-têtes et corps des message. La construction du classificateur se fait par apprentissage en ligne et actif. Les variantes de ce type de classificateur utilisent les mêmes types d'attributs mais extraits soit dans les en-têtes des messages, soit dans le corps.

L'autre classificateur est du type Bayésien Naïf. Les attributs sont des mots et des bi-mots extraits, pour la plupart, du corps des messages et uniquement dans quelques en-têtes : (`Subject`, `From`, `Content-Type`, `Content-Transfer-Encoding`, `User-Agent` et `X-Mailer`). La construction de ce classificateur est faite par apprentissage supervisée, *c.à.d.* tous les exemples sont utilisés.

La comparaison de deux algorithmes utilisant des représentations des données différentes n'a pas tellement de sens, néanmoins, la démarche reste intéressante puisque

On retrouve dans la Table 10.6 les valeurs des indicateurs de qualité de classement ($1 - AUC$ %, et taux d'erreur de classement par classe) évaluées sur trois périodes consécutives de 200 jours, pour chaque classificateur. On observe, selon le critère $1 - AUC$ %, une dégradation de la qualité de filtrage sur tous les classificateurs, mais plus prononcée pour le classificateur bayésien naïf. Néanmoins, si l'on regarde le taux d'erreur spécifique par classe, l'évolution dépend du type de classificateur.

La dérive dans dans l'ensemble des indicateurs est confirmée visuellement par les séries brutes (Figures 10.6, 10.13 et 10.14), en particulier la moyenne journalière de score pour le classificateur bayésien naïf. La perte d'efficacité peut s'expliquer par le mode d'apprentissage supervisée de ce classificateur qui tend à construire des classificateurs sur-ajustés. La Table 10.7 montre le nombre d'attributs et de messages retenus pour l'apprentissage de chacun des classificateurs.

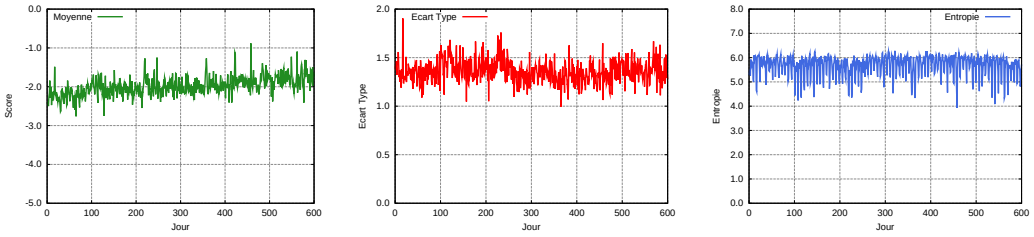
Vraisemblablement, des meilleurs résultats sont obtenus lorsque l'on utilise des informations aussi bien des en-tetes (méta-informations) et du corps des messages.

L'information intéressante que l'on peut retenir des variogrammes est que l'apparition de composantes périodiques dans l'écart-type du score est plus prononcée dans les classificateurs utilisant en priorité les en-têtes des messages. Le classificateur bayésien naïf présente une forte composante périodique dans la moyenne du score, ce qui est probablement dû au type d'attribut utilisé.

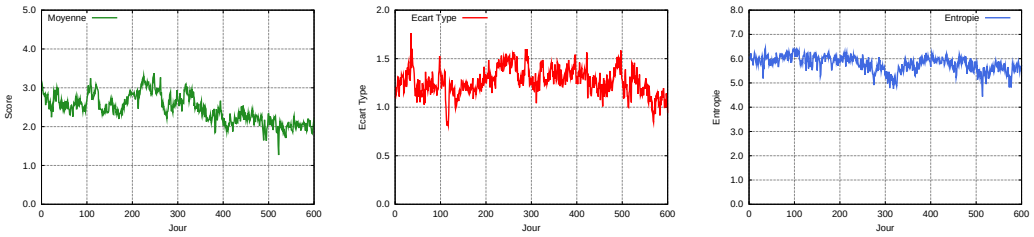
[9]Un *botnet* est un agent logiciel installé sur un ordinateur quelconque et qui exécute des tâches répétitives à la demande d'un ordinateur maître. Dans le domaine de la sécurité informatique, un réseau de botnets est un ensemble d'ordinateurs infectés par un logiciel malveillant (*e.g.* un virus), commandés par un maître et utilisés pour distribuer des spams, infecter autres ordinateurs (pour augmenter la taille du réseau), s'attaquer à des cibles précises ou toute autre activité malveillante.

	Classificateur *SLDC* - Corps et en-têtes des messages					
	Jour 1 - 200		Jour 201 - 400		Jour 401 - 600	
JM	0.0238 (0.018 - 0.032)	1.773 / 0.056	0.0489 (0.042 - 0.057)	2.438 / 0.120	0.1604 (0.144 - 0.179)	3.473 / 0.579
SB	0.0466 (0.038 - 0.057)	1.773 / 0.054	0.0355 (0.027 - 0.046)	2.438 / 0.042	0.0869 (0.074 - 0.102)	3.473 / 0.073
BG	0.1078 (0.091 - 0.128)	1.773 / 0.898	0.1337 (0.115 - 0.155)	2.438 / 0.526	0.3461 (0.309 - 0.388)	3.473 / 0.519
	Classificateur *SLDC* - Corps des messages					
	Jour 1 - 200		Jour 201 - 400		Jour 401 - 600	
JM	0.8382 (0.788 - 0.892)	7.975 / 1.519	0.9843 (0.928 - 1.044)	7.573 / 2.187	1.7968 (1.718 - 1.879)	7.712 / 4.282
SB	0.8859 (0.828 - 0.948)	8.010 / 1.453	1.2493 (1.188 - 1.314)	7.569 / 3.096	1.8163 (1.723 - 1.914)	7.712 / 3.835
BG	1.0562 (0.992 - 1.124)	8.010 / 2.029	1.1154 (1.055 - 1.179)	7.569 / 2.587	1.8314 (1.766 - 1.899)	7.712 / 4.334
	Classificateur *SLDC* - En-têtes des messages					
	Jour 1 - 200		Jour 201 - 400		Jour 401 - 600	
JM	0.0410 (0.031 - 0.055)	1.632 / 0.073	0.0711 (0.059 - 0.085)	2.700 / 0.286	0.2558 (0.232 - 0.282)	3.682 / 0.960
SB	0.0619 (0.046 - 0.083)	1.632 / 0.080	0.0452 (0.033 - 0.061)	2.700 / 0.041	0.1312 (0.112 - 0.154)	3.682 / 0.095
BG	0.0846 (0.068 - 0.105)	1.632 / 0.648	0.0867 (0.069 - 0.108)	2.700 / 0.295	0.2301 (0.199 - 0.266)	3.682 / 0.377
	Classificateur Bayésien Naïf					
	Jour 1 - 200		Jour 201 - 400		Jour 401 - 600	
JM	0.1721 (0.155 - 0.192)	0.339 / 3.965	2.0577 (2.005 - 2.112)	0.611 / 9.685	3.4842 (3.399 - 3.571)	0.783 / 23.75
SB	0.1948 (0.178 - 0.213)	0.339 / 7.125	1.3434 (1.299 - 1.389)	0.611 / 10.67	3.1534 (3.072 - 3.236)	0.783 / 29.72
BG	0.1507 (0.141 - 0.161)	0.339 / 8.280	3.3061 (3.241 - 3.372)	0.611 / 15.64	4.8255 (4.743 - 4.909)	0.783 / 31.09

TAB. 10.6: Dérive de l'efficacité de classement pour les quatre classificateurs. La durée de la période d'expérimentation (600 jours) a été partagée en 3 étapes de 200 jours. Dans chaque étape nous avons évalué les indicateurs *1-AUC%*, avec intervalles de confiance à 95 % et les taux d'erreur de classement (%) - faux positifs et faux négatifs.

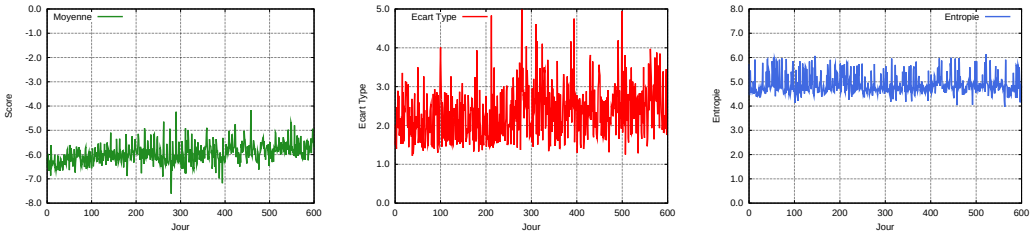


(a) Moyenne du score - JM-H (b) Écart-type du score - JM-H (c) Entropie du score - JM-H

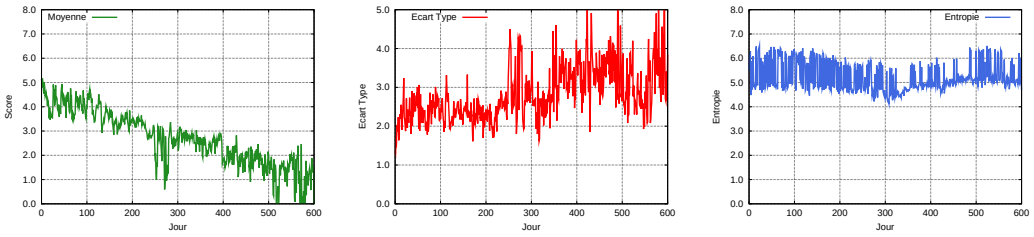


(d) Moyenne du score - JM-S (e) Écart-type du score - JM-S (f) Entropie du score - JM-S

FIG. 10.13: Séries temporelles des indicateurs des flots *JM-H* et *JM-S*, classificateur *SLDC*, avec des attributs pris uniquement le corps des messages. Ces séries sont à comparer avec celles de la Figure 10.6.

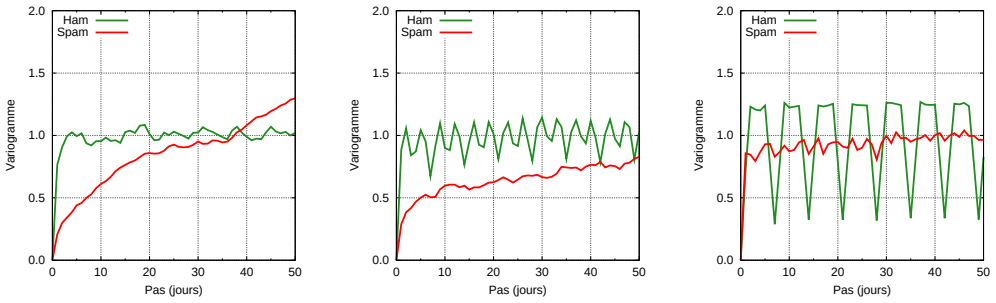


(a) Moyenne du score - JM-H (b) Écart-type du score - JM-H (c) Entropie du score - JM-H

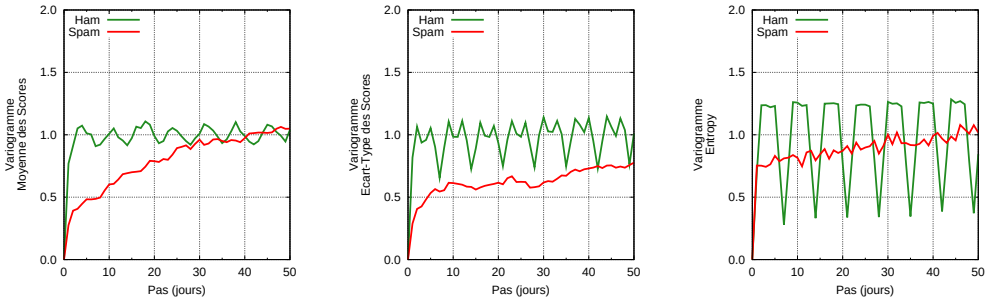


(d) Moyenne du score - JM-S (e) Écart-type du score - JM-S (f) Entropie du score - JM-S

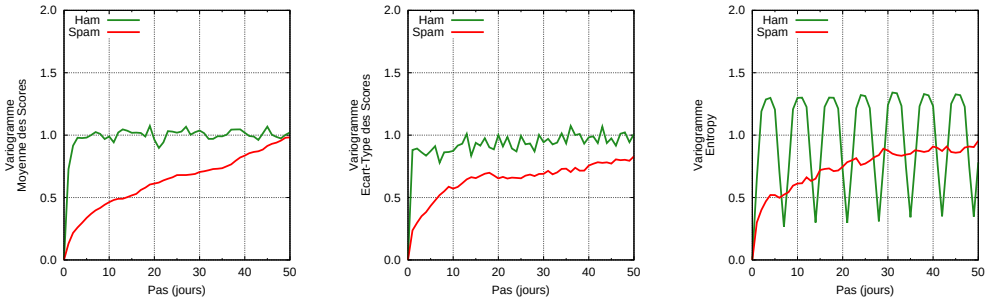
FIG. 10.14: Séries temporelles des indicateurs des flots *JM-H* et *JM-S* - classificateur Bayésien Naïf.



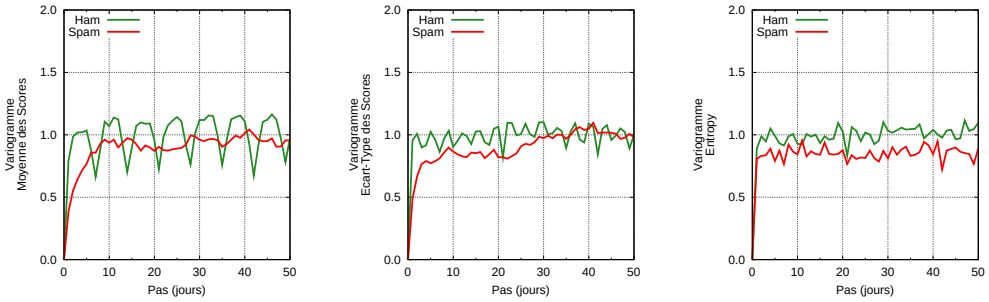
(a) Classificateur *SLDC* - en-têtes et corps des messages



(b) Classificateur *SLDC* - en-têtes des messages



(c) Classificateur *SLDC* - corps des messages



(d) Classificateur bayésien naïf

FIG. 10.15: Variogrammes des indicateurs des flots JM-H et JM-S - series brutes avec élimination de la tendance déterministe linéaire. Classificateur *SLDC* utilisant les en-têtes et le corps des messages (Figure 10.15a), seulement les en-têtes (Figure 10.15b), seulement le corps (Figure 10.15c) et classificateur bayésien naïf (Figure 10.15d).

Classificateur	Dimension	Spams	Hams
Bayésien Naïf	1505439	29093	8499
SLDC (en-têtes et corps)	204702	1043	570
SLDC (corps)	241562	3609	1951
SLDC (en-têtes)	125583	1656	837

TAB. 10.7: Complexité (dimension du vocabulaire) des classificateurs et nombre de messages retenus lors de la phase d'apprentissage. Le classificateur Bayésien Naïf a été construit avec apprentissage supervisé (et utilise tous les messages) tandis que les classificateurs *SLDC*, par apprentissage actif.

10.5 Conclusions

Dans ce chapitre nous avons collecté les résultats de classement de quatre flots de messages sur une période assez longue (20 mois). Le classificateur n'a pas été actualisé pendant toute la période d'expérimentation, de façon à mettre en évidence et observer la dérive des caractéristiques des flots. Nous avons construit des séries temporelles à partir des paramètres et indicateurs de classement. L'analyse de ces séries nous a permis de caractériser un certain nombre d'aspects de l'évolution temporelle de ces flots.

Nous avons tout d'abord constaté que les caractéristiques du flot de messages sont différentes et n'évoluent pas de la même façon selon la classe.

On observe que le nombre de messages par jour diminue pendant le weekend, de façon plus significative dans la classe des *hams* que des *spams*. Ces variations suivent le cycle d'activité professionnelle des utilisateurs de la messagerie et impactent les indicateurs de classement. Cet impact du nombre de messages par jour dans les indicateurs de classement de la classe *ham* suggère que, en plus de la dérive à long terme, les caractéristiques du flot de messages varie selon le jour de la semaine. Comme conséquence, le modèle probablement le plus adapté pour le flot de hams ne serait pas un modèle linéaire mais un modèle commuté, avec au moins deux régimes différents et un basculement progressif entre les régimes.

L'identification d'un processus à commutation de régime est intéressante à cause des implications que l'on trouvera dans un système de filtrage avec retour d'information (feedback), puisque ces retours n'étant jamais immédiats, il convient de vérifier le conséquences des retards lorsqu'ils couvrent les instants de changement de régime. L'influence de ces retards font partie des expérimentations du chapitre suivant.

L'existence d'une certaine ressemblance des indicateurs des flots de spam font croire qu'une partie significative de ces flots résulterait d'une combinaison linéaire d'un nombre, pas très élevé, de flots indépendants. Ces coefficients de mélange peuvent être (faiblement) variables dans le temps. Cette hypothèse n'est pas absurde puisqu'une fraction très importante de la quantité de spam en circulation (supérieure à 80 %) est générée par un petit nombre de sources (moins d'une centaine)[10].

L'unité d'agrégation que nous avons utilisé a été la journée, mais dans des expérimentations préliminaires, nous avons remarqué aussi des variations horaires dans le taux d'arrivée de messages. Nous n'avons pas pu étudier l'impact dans les indicateurs de classement, mais il est très probable que le résultat soit identique puisque la cause est identique. Une étude à cette échelle nécessite des flots de messages plus importants.

[10] *SpamHaus ROKSO* : `http://www.spamhaus.org/rokso/index.lasso`

CHAPITRE 11

Expérimentations de classement avec apprentissage actif en ligne

In theory there is no difference between theory and practice. In practice there is.

Yogi Berra

11.1 Introduction

Dans le chapitre précédent nous avons utilisé un classificateur, sans mise à jour, pour faire apparaître et étudier la dérive temporelle naturelle des flots de messages. Dans ce chapitre, le classificateur est mis à jour en continu, grâce au retour d'information de classement, et peut s'adapter aux changements des caractéristiques des flots. Des simulations sont faites dans différentes conditions de fonctionnement : retard et/ou bruit dans le retour d'information, apprentissage supervisé et plusieurs niveaux d'apprentissage actif.

Comme pour le chapitre précédent, nous utilisons un simple classificateur discriminant linéaire - *SLDC*. Les exemples sont présentés séquentiellement selon l'ordre chronologique réelle de leur réception lors de la constitution des corpus. L'apprentissage se fait en ligne grâce au retour d'information du classement correct des messages, soit suite à une demande faite par le classificateur (apprentissage actif), soit suite à une erreur de classement détectée par le destinataire final.

La contribution de ce chapitre constitue, à notre connaissance, la première expérimentation détaillée d'apprentissage actif, avec participation du destinataire, avec évaluation de l'impact des conditions opérationnelles et utilisant un corpus réel de messages.

11.2 Objectifs et Hypothèses

Dans ce chapitre nous étudions, empiriquement, les paramètres de fonctionnement d'un classificateur de messages électroniques, sur une assez longue durée, dans un contexte d'apprentissage actif et en ligne.

Pour cela, nous utilisons des ensembles de messages réels, aussi bien légitimes et du spam - des messages reçus par l'auteur et censés être significatifs d'une boîte aux lettres professionnelle. Malheureusement, à notre connaissance, il n'existe pas des corpus de messages légitimes, réels, réalistes et récents, disponibles pour des expérimentations, pour des raisons de confidentialité et respect de la vie privée.

Les évaluations de classificateurs de messages électroniques, que l'on trouve dans la bibliographie, utilisent généralement des ensembles synthétiques de messages - ce sont, pour la classe

spam, des messages en provenance de pièges à spam et, pour la classe *hams*, des messages en provenance de listes publiques de discussion ou de nouvelles, mais il n'est pas certain que ces ensembles de messages correspondent à des flots réels. De façon à pouvoir positionner nos résultats nous avons aussi utilisé des ensembles de messages, constitués selon les mêmes méthodes de ces évaluations, et effectué les expérimentations à la fois sur des ensembles réels et synthétiques. Cette comparaison n'est pas absolue, mais c'est la seule façon disponible de positionner notre modèle par rapport aux filtres de messages disponibles. Accessoirement, nous allons identifier des points qui font que l'utilisation d'ensembles synthétiques pour l'évaluation de classificateurs fondés sur l'apprentissage est hautement déconseillée.

En ce qui concerne les indicateurs opérationnels de classement, nous nous intéressons à la qualité de classement, exprimée par des taux d'erreur global et spécifique par classe, et aux conséquences de paramètres nuisibles tels le retard dans la boucle ou le niveau de bruit (qui représente les erreurs d'appréciation, par les destinataires). D'autres paramètres intéressants sont la complexité du classificateur à la fin de l'expérimentation, représentée par le nombre d'attributs, et le nombre de messages retenus pour l'apprentissage.

11.3 Modèle fonctionnel et contexte de simulation

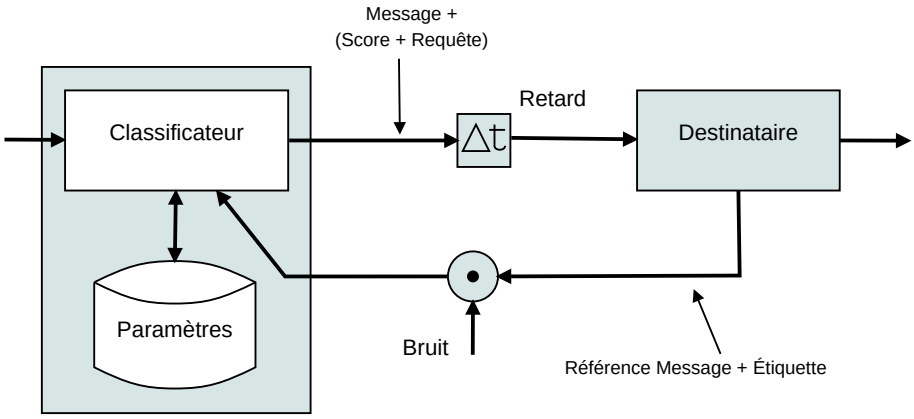


FIG. 11.1: Modèle fonctionnel utilisé pour la simulation de classement en ligne de messages électroniques avec apprentissage actif.

L'étude présentée ici est basée sur le modèle fonctionnel de la Figure 11.1, que nous avons simulé à l'aide d'un simulateur à événements discrets.

Les corpus utilisés (décrits plus loin) sont constitués de messages étiquetés avec la classe d'appartenance et la date, réelle, de leur arrivée.

La simulation de l'opération en ligne du classificateur est coordonnée par un ordonnanceur, qui scrute une pile d'événements triés par ordre chronologique et soumet chaque événement au module concerné. Les événements sont de deux types : l'arrivée d'un message (événement *classement*) et le retour d'une information de classe associé à un message (événement *apprentissage*). Les événements du premier type sont des événements externes imposés au simulateur, tandis que ceux du deuxième type résultent du fonctionnement même du simulateur.

11.3.1 Les modules du simulateur

Le simulateur a deux modules qui correspondent à des composants réels : le *classificateur* et le *destinataire* des messages, deux modules auxiliaires (*Retard* et *Bruit*) et un module d'*ordonnancement* dont dépend le fonctionnement du simulateur lui-même.

Le module Ordonnanceur - qui n'apparaît pas dans ce schéma, est chargé de la gestion de la simulation et de la communication entre les modules. Il scrute la pile d'événements dans l'ordre chronologique et soumet des tâches à chaque module selon l'événement à traiter ou le résultat du traitement effectué par les modules.

Le processus entier de simulation est déclenché par une suite d'événements, lus par l'ordonnanceur à partir d'un fichier, et qui correspondent à l'arrivée des messages. Ce fichier a le format suivant :

```
1249233003   spam    joe-work/joe-spam/S-0908/msg.0019947
1249233004   ham     joe-work/joe-ham/H-0908/msg.0000110
1249233004   spam    joe-work/joe-spam/S-0908/msg.0000958
1249233603   spam    joe-work/joe-spam/S-0908/msg.0000959
```

où la première colonne correspond à la date d'arrivé du message (seconds depuis *Unix epoch* : 1er/janvier/1970 00h00m00s), la deuxième colonne contient l'étiquette du message (information qui n'est utilisée que par le destinataire) et la dernière colonne contient le chemin du fichier avec le message à traiter.

Le Classificateur - est basé sur un algorithme discriminant linéaire - *SDLC* (voir description détaillée dans le Chapitre 9) - et fonctionne en deux modes : *classement* et *apprentissage*.

Le mode *classement* est déclenché par un événement d'arrivée d'un message. Le classificateur examine le message et lui attribue un score : le produit interne de la représentation du message et le vecteur de paramètres du classificateur. La valeur neutre du score (0) est utilisée comme seuil (trivial) de classement : au dessus de cette valeur la classe *spam* est attribuée au message et au dessous c'est la classe *ham*. Ce score correspond, en principe, au *logit* de la probabilité que le message en cours appartienne à classe spam. Le score en échelle de probabilité $[0, 1]$ est aussi disponible.

L'apprentissage actif (voir description détaillée dans la Section 5.4.3 et Chapitre 9) implémenté dans le simulateur est basé sur la *marge de classement* définie, par :

$$marge = |score - 0.5| \tag{11.1}$$

et qui est un indicateur de l'incertitude de classement. Si la marge est inférieure à une valeur prédéfinie : le *seuil de requête*, le classificateur active un drapeau, transmis au destinataire avec le message lui indiquant que le classificateur souhaite connaître le classement correct du message. À noter que si le seuil est égal à 0.5 (valeur maximale), alors tous les messages sont intégrés dans le processus d'apprentissage, ce qui correspond à la situation d'*apprentissage supervisé*. La valeur extrême opposée - seuil nul - correspond à la situation où le classificateur n'est jamais mis à jour.

Le mode *apprentissage* est déclenché lorsque le destinataire retourne la bonne étiquette associée à un message. Le classificateur évalue, à nouveau le score et la marge - ceci est nécessaire puisque le classificateur a pu être mis à jour après le premier passage du message. Si la nouvelle marge est inférieure au *seuil de requête*, le classificateur adapte les corrige paramètres de l'algorithme de classement, selon la méthode décrite dans le Chapitre 9 et Annexe C (approximation stochastique). La vitesse d'apprentissage suit la règle :

$$\eta_i = \frac{1 - \eta_\infty}{i} + \eta_\infty,\ i > 0,\ 0 \leq \eta_\infty < 1 \tag{11.2}$$

La valeur asymptotique de η_i ne doit pas être nulle, de façon à ce que le classificateur puisse s'adapter aux changements dans le flot et défini *"l'inertie"* du classificateur. Dans des expérimentations préliminaires nous n'avons pas trouvé des différences significatives pour des valeurs asymptotiques dans l'intervalle $[0.001, 0.004]$. En dehors de cette plage, les valeurs plus hautes rendant le classificateur trop sensible au bruit et les valeurs plus basses font qu'il ne réagit pas assez vite aux changements. Dans l'ensemble de ces expérimentations, nous avons fixé $\eta_\infty = 0.002$. Cette valeur est compatible avec la *"thumb rule"* indiquée par Benveniste et all [18, p. 150] : autour de un millième.

Corpus	Classe	Nombre	Description
JM-H	Ham	86858	Les messages légitimes reçus par l'auteur
JM-S	Spam	379210	Les spams reçus par l'auteur
SB-S	Spam	486142	Les spams reçus dans un piège à spam, géré par l'auteur, dans un sous-domaine de `ensmp.fr`
BG-S	Spam	1219031	Les spams reçus dans un piège à spam géré par Bruce Guenter, et disponible à `http://untroubled.org/spam/`
SYNT-H	Ham	196112	Un corpus synthétique constitué de messages en provenance des archives publiques d'environ 200 listes de diffusion (en français et en anglais) ainsi que des abonnements à des services de nouvelles scientifiques ou d'actualités (ACM, Science Direct, CNN, CBS, Foxnews, NewYork Times, ...)

TAB. 11.1: Ensembles de messages utilisés dans les expérimentations. Les ensembles JM-H et JM-S sont constitués avec des messages envoyés à des personnes réelles. Les autres ensembles sont des ensembles "synthétiques" constitués, selon la catégorie, par des messages envoyés à des pièges à spam ou des messages distribués par des listes de diffusion.

Le Destinataire - joue un rôle trivial : si le drapeau de demande de classement correct est actif ou si le classement du message est erroné, alors il retourne le message vers le classificateur à l'aide d'un événement *apprentissage*.

Le module "Retard" - est un artifice permettant de simuler le délai entre le moment où le classificateur a traité le message et le moment où le message a été lu par le destinataire. Dans le cas où le destinataire doit retourner une information de classement, un événement correspondant est ajouté à la pile d'événements, pour la date $t_i+\Delta_i$. Deux types de retard ont été expérimentés : un retard constant et un retard aléatoire de distribution exponentielle avec un seuil de 10 minutes. Dans la réalité, le retard doit suivre des lois plus complexes, en particulier dépendre du moment de la journée et du jour de la semaine. A notre connaissance, il n'y a pas de travaux publiés sur ce sujet.

Le module "Bruit" - , comme le module précédent, permet de simuler les erreurs d'appréciation du classement correct des messages, par le destinataire. Le passage par ce module fait que l'information de classe retournée par le destinataire est inversée aléatoirement selon une valeur générée aléatoirement avec distribution de Bernouilli dont le paramètre est la probabilité d'erreur à simuler. Il s'agit d'un bruit dit uniforme dans le sens où il est indépendant des caractéristiques de chaque message.

11.3.2 Corpus de messages

Nous avons utilisé les mêmes corpus de messages déjà utilisés dans le Chapitre 10 : *JM-H*, *JM-S*, *SB-S* et *BG-S*. Les deux premiers correspondent aux messages des deux catégories reçus par une même personne, réelle, et à un même endroit, tandis que les deux derniers, ce sont des messages envoyés à des pièges à spam (Voir Table 11.1). Nous avons ajouté un dernier ensemble de messages, *SYNT-H*, construit à partir d'archives de listes de discussion publiques, dont la plupart typiques de la communauté d'enseignement supérieur et de recherche, et de listes de diffusion de nouvelles scientifiques ou actualités - ce sont des messages en français et en anglais. Il s'agit, pour les cinq ensembles, de messages collectés dans la période allant du mois de janvier 2009 à août 2010.

Les ensembles de messages ont été combinés par pairs (un de chaque classe) et triés par ordre chronologique de la date de réception. Les deux couples les plus significatifs sont *JM-H/JM-S* et *SYNT-H/BG-S*. Le premier représente les messages reçus par une personne réelle et le deuxième est, typiquement, le type de corpus utilisé dans les évaluations de filtres anti-spam par les revues spécialisées (*e.g.* Virus Bulletin - VBSPAM[1]) et nous permet de positionner notre expérimentation par rapport aux résultats de ces évaluations.

Nous avons aussi utilisé le corpus public de messages TREC-07 [57], ce qui nous permet d'étalonner et comparer nos résultats avec ceux obtenus par le classificateur similaire proposé par Gordon Cormack [58]. Accessoirement, il s'agit d'un corpus synthétique dans le sens où il a été constitué exclusivement pour le workshop Spam de *TREC-2007*, mais c'est aussi le premier corpus public constitué des deux classes de messages collectés sur un même point. Par contre, son étendue temporelle ne coïncide pas avec celle des autres ensembles.

Pour simuler le fonctionnement d'un filtre en bordure d'un domaine, les messages ont subi un traitement de façon à enlever les entêtes rajoutés à l'intérieur du domaine, des informations qui ne sont pas, à priori, discriminantes de la classe du message. En plus, dans le cas des ensembles dits synthétiques, certains entêtes du être modifiés de façon à ce qu'ils puissent être vus, par le filtre, comme des messages destinés à des destinataires réels, les mêmes de la classe opposée. Les raisons et la pertinence de ces derniers traitements seront discutés en même temps que les résultats.

11.3.3 Protocole de simulation

Dans les expérimentations de classement en boucle ouverte (Chapitre 10), nous avons utilisé un classificateur pré-construit avec les messages reçus pendant les deux mois avant la période en étude, et aucune mise à jour n'a été appliquée pendant l'expérimentation. Ici, nous partons d'un classificateur vide qui sera construit au long de l'expérimentation.

On peut distinguer deux phases d'apprentissage : une phase initiale transitoire (dite *d'accrochage*) et une phase dite de *poursuite*, où le but de l'apprentissage est l'adaptation du classificateur aux changements (lents ou mineurs) des caractéristiques du flot.

Nous n'avons pas tenu compte, dans les résultats de synthèse, des messages des 30 premiers jours (choix arbitraire), de façon à ce que résultats ne tiennent pas compte de la phase transitoire. Néanmoins, les résultats des premiers jours sont toujours enregistrés pour pouvoir étudier la durée et le comportement du classificateur pendant la phase d'accrochage.

Pour chaque couple d'ensemble de messages nous avons fait varier :

- le retard dans le retour de l'information de classement correct, par le destinataire. Nous avons effectué des simulations avec retard fixe et retard aléatoire avec distribution exponentielle ;
- le taux d'erreur dans le retour d'information de classement ;
- la marge utilisée dans l'apprentissage actif, la faisant varier dans l'intervalle $[0.05, 0.50]$. La borne supérieure correspond à une situation d'apprentissage supervisée où tous les exemples sont utilisés.

Pour chaque contexte de simulation défini par les paramètres décrits ci-dessus, nous avons observé :

- Taux d'erreur de classement spécifique par classe et global représenté par l'aire sous la courbe ROC (1-AUC%) ;
- Taux de requêtes (query rate) la fraction du nombre de messages dont le classificateur demande la classe d'appartenance (l'étiquette). Ce paramètre est intéressant puisqu'il indique le niveau de participation du destinataire dans le classement des messages ;
- Taux d'apprentissage - la fraction du nombre de messages classés qui ont été effectivement utilisés pour l'apprentissage ;
- Dimension du classificateur à la fin de l'expérimentation.

[1] Virus Bulletin `http://www.virusbtn.com/vbspam/index`

Ces (nombreuses) mesures ont été faites avec les tous les ensembles de messages. Nous présentons, dans la section suivante, les résultats globaux pour les différents couples étudiés et, ensuite, les résultats détaillés pour chaque paramètre pour le couple *JM-H/JM-S*.

11.4 Résultats

11.4.1 Les différents flots de messages

Le Tableau 11.2 synthétise les résultats obtenus avec les différentes combinaisons possibles de couples d'ensembles *hams/spams*, *réels/synthétiques* de messages, obtenus avec une marge d'apprentissage actif fixée à 0.35 et conditions idéales de retour d'information : pas de retard et pas de bruit.

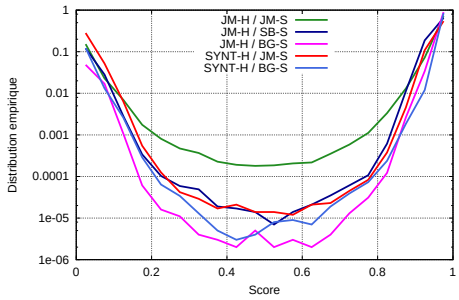


FIG. 11.2: Les histogrammes normalisés (en échelle logarithmique), estimés sur 20 bins, des scores des messages (en échelle de probabilité) montrent que le nombre de messages à l'intérieur de la région d'apprentissage actif est bien plus importante lorsque les deux classes sont constituées avec de messages réels envoyés à des destinataires physiques.

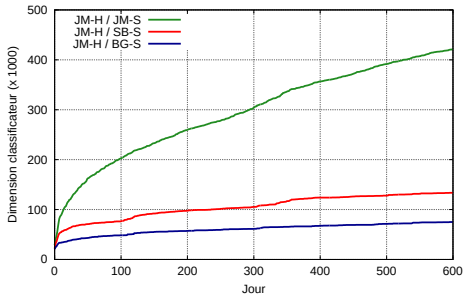


FIG. 11.3: Evolution de la dimension (en nombre d'attributs) du classificateur. On remarque la croissance plus rapide pour le couple *JM-H/JM-S*, qui s'explique par le nombre plus important de messages dans la région d'apprentissage actif.

Les résultats de chaque couple d'ensembles de messages sont présentés en deux lignes : la première ($\Delta_t = 0$) tient compte de la durée entière de l'expérimentation tandis que dans la deuxième ($\Delta_t = 30$) le début d'acquisition est décalé de 30 jours, un choix arbitraire, de façon à ne pas tenir compte de la phase transitoire d'initialisation du classificateur.

On remarque une différence importante entre les résultats des deux couples extrêmes : *JM-H/JM-S* (ensembles de messages réels) et *SYNT-H/BG-S* (ensembles de messages synthétiques), avec des résultats nettement meilleurs pour ce dernier et des résultats intermédiaires pour des combinaisons mixtes.

Corpus	Δt Jour	**Classe**	**Messages** Nb	**Erreurs** Nb	**Erreurs** %	**1-AUC%** %	**1-AUC%** Intervalle de Confiance	**Requêtes** Nb	**Requêtes** %	**Dim.** Attrs
JM-H	0	ham	86858	285	3.281e-01	3.634e-03	(2.681e-03 - 4.926e-03)	2060	2.372	422696
		spam	379210	122	3.217e-02			2667	7.033e-01	
JM-S	30	ham	82158	208	2.532e-01	3.248e-03	(2.316e-03 - 4.556e-03)	1672	2.035	
		spam	363198	109	3.001e-02			2051	5.647e-01	
JM-H	0	ham	86858	13	1.497e-02	3.372e-06	(1.837e-06 - 6.190e-06)	341	3.926e-01	133893
		spam	486142	17	3.497e-03			495	1.018e-01	
SB-S	30	ham	82158	2	2.434e-03	1.471e-06	(6.186e-07 - 3.497e-06)	221	2.690e-01	
		spam	466714	16	3.428e-03			267	5.721e-02	
JM-H	0	ham	86858	5	5.757e-03	1.577e-07	(4.760e-08 - 5.226e-07)	136	1.566e-01	75394
		spam	1219031	23	1.887e-03			247	2.026e-02	
BG-S	30	ham	82158	3	3.652e-03	6.746e-08	(8.724e-09 - 5.217e-07)	73	8.885e-02	
		spam	1172805	21	1.791e-03			143	1.219e-02	
SYNT-H	0	ham	196112	10	5.099e-03	1.315e-06	(4.594e-07 - 3.765e-06)	456	2.325e-01	139495
		spam	379210	14	3.692e-03			335	8.834e-02	
JM-S	30	ham	184134	0	0.000	1.914e-07	(3.342e-08 - 1.096e-06)	293	1.591e-01	
		spam	363198	12	3.304e-03			189	5.204e-02	
SYNT-H	0	ham	196112	12	6.119e-03	6.030e-06	(1.972e-06 - 1.844e-05)	586	2.988e-01	172627
		spam	1219031	33	2.707e-03			578	4.741e-02	
BG-S	30	ham	184134	1	5.431e-04	2.940e-06	(2.663e-07 - 3.245e-05)	407	2.210e-01	
		spam	1172805	30	2.558e-03			389	3.317e-02	

TAB. 11.2: Synthèse des résultats de classement avec apprentissage actif et en ligne, sur une période de 600 jours, dans des conditions idéales : rétroaction non bruitée et sans retard. Chaque couple présente deux ensemble de mesures : avec ($\Delta t = 0$) ou sans ($\Delta t = 60$) inclusion de la période transitoire. Ces résultats sont à comparer avec ceux obtenus avec un classificateur en boucle ouverte (Table 10.1).

Corpus	Messages (%)
JM-H/JM-S	0.999 %
JM-H/SB-S	0.146 %
JM-H/BG-S	0.0279 %
SYNT-H/JM-S	0.137 %
SYNT-H/BG-S	0.0804 %

TAB. 11.3: Fraction du nombre de messages dont le score de classement se situe à l'intérieur de la région de classement actif : $[0,15, 0,85]$ (Table 11.2), bien plus importante dans de cas des ensembles de messages réels.

Corpus	Messages (%)
JM-H/JM-S	0.999 %
JM-H*/JM-S	0.854 %
JM-H/JM-S*	0.952 %
JM-H*/JM-S*	0.823 %

TAB. 11.4: Fraction du nombre de messages du couple *JM-H/JM-S* dont le score de classement se situe à l'intérieur de la marge de classement actif : $[0,15, 0,85]$ (Table 11.5). Les messages publicitaires ont été enlevés des ensembles marqués avec un '*'.

Les messages dans la zone d'apprentissage actif

Si l'on observe la distribution des scores des messages (Figure 11.2 et Table 11.3), on remarque que la fraction de messages dont le score (en probabilité) se trouve à l'intérieur de la plage d'apprentissage actif ($[0.15, 0.85]$) est plus importante dans les le couple *JM-H/JM-S* que dans les autres couples. Cet indicateur représente l'incertitude de classement.

Dans le couple *JM-H/JM-S*, la plupart des messages mal classés (74% des spams et 87% des hams) et dont le score se trouve dans la plage d'apprentissage actif sont des messages publicitaires particulières. Il s'agit de messages envoyés par des entreprises de marketing connues et légitimes distribuant, à la fois, de la publicité abusive (donc des *spams*) et des *newsletters* ou des messages commerciales de partenaires connus (donc des *hams*) - des messages très similaires sur les deux classes.

Pour examiner l'influence de cette catégorie de messages dans les résultats de classement, nous avons comparé les résultats de l'ensemble *JM-H/JM-S* avec ceux que l'on obtient lorsque l'on enlève ce type de messages (Table 11.5), et nous constatons que leur suppression améliore légèrement la qualité de classement. Curieusement, l'amélioration dans les taux d'erreur est plus effective lorsque la suppression se fait dans la classe *spam*, alors que dans le nombre de requêtes et dimension finale du classificateur, l'amélioration est plus effective quand la suppression se fait dans la classe *ham*. On remarque notamment une notable diminution dans la dimension du classificateur lorsque ce type de message est enlevé du flot.

La Table 11.4 montre les fractions de messages à l'intérieur de la plage d'apprentissage actif après suppression des messages publicitaires. Aussi, on remarque que la fraction de messages de cette catégorie n'est pas la même dans les deux classes : $1,43\%$ des hams et $3,97\%$ des spams. Hormis l'impact certain de cette catégorie de messages dans l'efficacité de classement, il ne semble pas judicieux de généraliser les observations restantes.

Ces messages publicitaires "légitimes" n'apparaissent pas, ou très rarement, dans les ensembles

Corpus	Δt Jour	**Classe**	**Messages** Nb	**Erreurs** Nb	%	**1-AUC%** %	Intervalle de Confiance	**Requêtes** Nb	%	**Dim.** Attrs
JM-H	0	ham	86858	285	3.281e-01	3.634e-03	(2.681e-03 - 4.926e-03)	2060	2.372	422696
		spam	379210	122	3.217e-02			2667	7.033e-01	
JM-S	30	ham	82158	208	2.532e-01	3.248e-03	(2.316e-03 - 4.556e-03)	1672	2.035	
		spam	363198	109	3.001e-02			2051	5.647e-01	
JM-H	0	ham	86858	243	2.798e-01	2.643e-03	(1.841e-03 - 3.794e-03)	1884	2.169	403743
		spam	364148	88	2.417e-02			2454	6.739e-01	
JM-S*	30	ham	82158	167	2.033e-01	2.326e-03	(1.517e-03 - 3.566e-03)	1505	1.832	
		spam	348468	74	2.124e-02			1850	5.309e-01	
JM-H*	0	ham	85618	228	2.663e-01	3.260e-03	(2.071e-03 - 5.130e-03)	1775	2.073	385315
		spam	379210	100	2.637e-02			2263	5.968e-01	
JM-S	30	ham	80971	166	2.050e-01	2.920e-03	(1.868e-03 - 4.564e-03)	1419	1.752	
		spam	363198	86	2.368e-02			1717	4.727e-01	
JM-H*	0	ham	85618	207	2.418e-01	2.329e-03	(1.634e-03 - 3.320e-03)	1646	1.922	366244
		spam	364148	83	2.279e-02			2105	5.781e-01	
JM-S*	30	ham	80971	143	1.766e-01	2.060e-03	(1.357e-03 - 3.129e-03)	1293	1.597	
		spam	348468	69	1.980e-02			1569	4.503e-01	

TAB. 11.5: Résultats de classement du l'ensemble *JM-H/JM-S*. Les messages publicitaires ont été enlevés des ensembles suffixés par un "-*".

synthétiques, que ce soient des messages en provenance de pots de miel ou des listes de diffusion. En effet, la présence de ce type de message, à cheval sur les deux classes, rend les classes moins facilement séparables. Cette catégorie de message est parfois désignée par l'expression *message gris* (ou *gray mail*, en anglais). Leur présence, dans les deux classes affecte le taux d'erreurs de classement, le nombre de messages utilisés pour l'apprentissage actif et surtout la dimension du classificateur construit, comme le montre l'évolution temporelle de la dimension du classificateur - Figure 11.3 - bien plus importante dans le flot de messages réels (JM-H/JM-S) que dans les autres. On peut penser que, à long terme, ces messages peuvent nuire l'efficacité de classement, et que ce type de message mérite une attention ou un traitement particulier.

Autres différences

Néanmoins, l'amélioration que nous avons obtenu n'est pas suffisante pour expliquer la différence trouvée. Nous avons mentionné, dans la Section 11.3.2, que les ensembles de messages ont subi des traitements de façon à ce que, aux yeux du classificateur, ils puissent ressembler à des messages collectés tous aux même endroit et vers des destinataires du même domaine local. En effet, sans cela, le classificateur fini par classer les messages par leur origine et non pas par les catégories voulues (*ham* et *spam*).

Cette hypothèse se confirme lorsque l'on consulte les attributs les plus discriminants du classificateur (les attributs de poids les plus forts) : le nom de domaine, le modèle et version du logiciel sur le serveur ou encore des entêtes spécifiques. Cette remarque est valable en particulier pour les classificateurs avec apprentissage en ligne. Le traitement automatisé des entêtes peut devenir assez complexe, en particulier lorsque le type de logiciel de messagerie installé dans les points de collecte ne sont pas identiques ou alors dans certaines configurations particulières. Cette complexité peut finir par détériorer l'efficacité de filtrage des ensembles réels. Ainsi, il ne nous semble pas utile de consacrer trop d'effort sur cela et considérer que les évaluations de classificateurs basés sur l'apprentissage en ligne ne doit être faite qu'avec des ensembles de messages collectés en même temps et au même endroit.

Reste une hypothèse que nous n'avons pas pu vérifier, et qui concerne la diversité des messages. On a bien constaté que, pour la catégorie des spams, on retrouve, grosso modo, les mêmes messages aussi bien dans les ensembles collectés dans des pièges à spam ou envoyés à un destinataire réel. Néanmoins, pour les messages légitimes synthétiques, nous avons utilisé un nombre limité (~ 200) de listes de diffusion ou de nouvelles, envoyés par un nombre petit d'expéditeurs (< 20). Or, ces messages sont formatés de façon assez uniforme avec des sujets et vocabulaire assez stables. La vérification de cet hypothèse nécessite d'autres outils et surtout d'avoir accès à l'ensemble des messages (*hams* et *spams*) reçus pendant une période assez longue, ce qui nous semble assez difficile pour des raisons pratiques et de protection de la vie privée et de confidentialité.

Vitesse d'apprentissage

Si bien que l'idée, qualitative, de la vitesse d'apprentissage soit intuitive - *la rapidité avec laquelle un dispositif s'adapte à son environnement* - les définitions peuvent varier selon le contexte. Nous faisons le choix arbitraire de nous inspirer du "temps de montée" (*rise time*) [2] que l'on définit, en électronique, comme le délai de basculement d'un signal, entre 10% et 90% de l'amplitude du changement, déduits les éventuels pics de dépassement *(overshoot)*.

Le signal qu'il convient d'observer est le score des messages, le paramètre de régulation du classificateur. En fait nous utilisons le score moyen journalier. Du point de vue de l'efficacité de filtrage il conviendrait plutôt de observer l'erreur de classement. Nous allons regarder les deux.

Comme dans les analyses déjà faites, nous avons séparé, par classe, les résultats de score moyen journalier et les présentons dans les Figures 11.4a et 11.4b. On observe que tous les indicateurs dépassent déjà, dès le premier jour, la moitié de la plage de variation. Pour évaluer

[2] `http://www.atis.org/glossary/definition.aspx?id=2014`

le "temps de montée", nous avons considéré comme valeur cible, la moyenne du score pendant la période d'expérimentation, excluant les 120 premiers jours (pour être surs de ne pas inclure la phase transitoire) et pris le nombre de jours nécessaires pour atteindre pour la première fois 90% de cette valeur. Les résultats obtenus sont présentés dans la Table 11.6.

	Ham		Spam	
	Δ_{score}	t_r *(jours)*	Δ_{score}	t_r *(jours)*
JM-H/JM-S	-4.39	19	4.27	30
JM-H/SB-S	-3.70	22	3.48	37
JM-H/BG-S	-3.37	19	4.22	30

TAB. 11.6: Temps de montée (rise time), en jours, du score moyen des messages, par classe. Cet indicateur évalue le temps pris pour que le score moyen journalier effectue 80% de la variation attendue (10% à 90%) : passant de la valeur initiale à la valeur moyenne finale.

L'autre point de vue est l'efficacité de filtrage, *c.à.d.* le taux d'erreur. Une évaluation directe par le nombre de messages classés en erreur par jour n'est pas une mesure significative puisque le nombre d'échantillons (~ 1000) est faible pour les taux d'erreur en jeu (entre 0.01% et 0.5%).

Une approche alternative consiste à supposer que, à chaque jour, le score des messages suit une distribution normale. Cette hypothèse est plausible puisque le score résulte d'une somme de variables aléatoires. On peut alors déterminer la borne de seuil qui correspondant à un taux d'erreur fixe.

Dans une approche alternative, émettons l'hypothèse que, pour chaque jour, le score des messages, de chaque classe, est une variable aléatoire de distribution normale :

$$\begin{array}{c} \mathcal{N}(\mu_h(i), \sigma_h(i)),\ \mu_h(i) < 0 \\ \mathcal{N}(\mu_s(i), \sigma_s(i)),\ \mu_s(i) > 0 \end{array} \tag{11.3}$$

On peut alors déterminer une borne de score correspondant à un taux d'erreur fixe donnée comme, par exemple, 2.5 % (référence usuelle lorsqu'on utilise des distributions normales).

Ainsi, pour chaque jour i et classes *ham* et *spam*, les bornes de score qui font que les probabilités d'erreur spécifique soient inférieures à 2.5% sont définies par :

$$\begin{array}{c} P_h(score > \hat{\mu}_h(i) + 1.96\ \hat{\sigma}_h(i)) < 0.025 \\ P_s(score < \hat{\mu}_s(i) - 1.96\ \hat{\sigma}_s(i)) < 0.025 \end{array} \tag{11.4}$$

Ces bornes sont présentées dans les Figures 11.4c et 11.4d montrent que, à l'exception des hams du couple de messages *JM-H/JM-S*, le taux d'erreur spécifique de classement est déjà inférieur à 2.5%.

Le test de normalité de *Cramer-Von Mises*[3], appliqué sur quelques dizaines de jours échantillons choisis au hasard a montré que l'hypothèse de normalité n'est pas vérifiée. Nous avons évalué, pour chaque jour, les troisième et quatrième moments sans dimension (*Skewness* et *Kurtosis*) (voir, p.ex. l'Annexe B ou [75]) des distributions du score des messages. Ce sont des indicateurs de l'asymétrie et de l'aplatissement (ou "*pointicité*") d'une distribution[4].

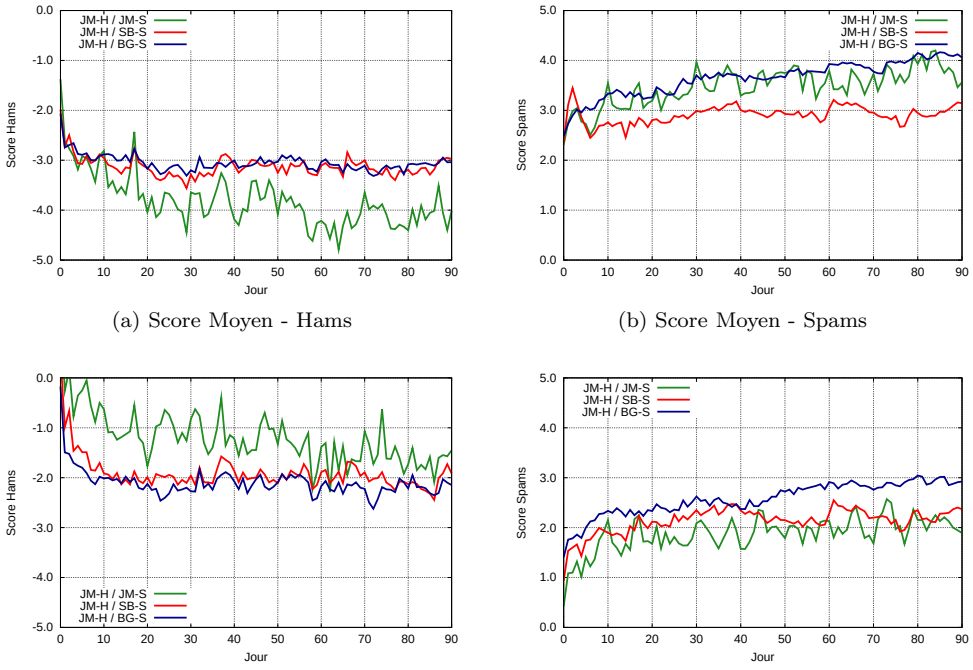
Les valeurs de ces indicateurs, synthétisées dans le Tableau 11.7, ne permettent pas de tirer des conclusions généralisables, mais étant donné qu'elles restent majoritairement faibles on peut penser que l'approximation reste acceptable, même si l'hypothèse de normalité n'est pas respectée.

[3]Le test statistique d'ajustement de *Cramer-Von Mises* a été effectué par le logiciel R.

[4]Cette définition de *Kurtosis* est souvent appelée *Kurtosis en excès*, à cause de la valeur 3 déduite, permettant une comparaison avec une distribution normale.

			Min.	1st Qu.	Median	3rd Qu.	Max.	Mean
JM-H	Ham	skewness	-2.1430	-0.5574	-0.2920	-0.0802	1.3380	-0.3181
		kurtosis	-1.2950	-0.2475	0.2698	1.0490	6.1230	0.4814
JM-S	Spam	skewness	-1.3890	-0.1730	0.0125	0.1670	0.8549	-0.0158
		kurtosis	-1.1950	-0.1839	0.0886	0.5123	11.3300	0.3683
JM-H	Ham	skewness	-2.3300	-1.0450	-0.6630	-0.3061	1.2120	-0.6566
		kurtosis	-1.4960	0.2836	1.1460	2.2290	9.3040	1.4030
SB-S	Spam	skewness	-1.5270	-0.2140	0.1258	0.4029	1.1030	0.0975
		kurtosis	-1.3320	-0.2962	0.0885	0.5716	14.8600	0.2384
JM-H	Ham	skewness	-1.5580	-0.7237	-0.5180	-0.2772	1.2290	-0.4784
		kurtosis	-1.5500	0.0553	0.4460	0.9687	8.0000	0.5990
BG-S	Spam	skewness	-1.7510	-1.0720	-0.8557	-0.5834	0.3841	-0.7918
		kurtosis	-0.8411	0.3818	1.0840	1.7920	7.6210	1.1630

TAB. 11.7: Synthèse des troisième et quatrième moments (*kurtosis* et *skewness*) de la distribution empirique du score des messages, par jour.



(a) Score Moyen - Hams

(b) Score Moyen - Spams

(c) Borne supérieure - Hams

(d) Borne inférieure - Spams

FIG. 11.4: Évolution de la moyenne journalière du score (en échelle *logit*) des messages pendant la période d'accrochage. Le score des messages est le paramètre de régulation de la boucle de rétroaction. On constate que ce paramètre s'approche très rapidement de la valeur en régime permanent.

Comparaison avec les résultats de TREC-07

Le classificateur que nous avons utilisé dans nos expérimentations est basé sur celui proposé par Goodman et Yih [115], et repris par Cormack [58]. Du point de vue conceptuel, nous utilisons un taux d'apprentissage variable et décroissant (approximation stochastique), alors que ce paramètre est fixe dans les versions précédentes. Les autres différences relèvent de l'implémentation : Cormack utilise le message brut, alors que nous excluons quelques entêtes des messages et nous n'utilisons pas les messages bruts, mais nous prenons soin de décoder les parties MIME (multimédia).

Lors du workshop TREC-07 SPAM [57], les meilleurs résultats ont été obtenus par le classificateur à SVM proposé par Sculley [230] et le classificateur à Régression Logistique utilisé par Cormack [58].

Dans la Table 11.8 nous comparons nos résultats (apprentissage actif) avec ceux obtenus par Cormack et Sculley sur le corpus TREC07 [57] dans les tâches d'apprentissage actif (avec la contrainte de ne pas utiliser plus de 1000 exemples) et d'apprentissage supervisé.

Pour l'apprentissage actif, nos résultats sont légèrement meilleurs, selon l'indicateur 1-AUC%. Si cette différence n'est pas suffisamment significative, elle indique néanmoins que l'efficacité de notre classificateur est au moins comparable à celle de l'état de l'art actuellement connus.

Nous avons ajouté les résultats des solutions proposées par Cormack et Sculley, dans un contexte d'apprentissage supervisé. Dans ce contexte, leurs résultats sont meilleurs que les nôtres en apprentissage actif. Néanmoins, ce contexte est irréaliste puisque, d'une part, il exige l'information d'étiquette de tous les messages vus et d'autre part, il est fort probable que ces classificateurs soient sur-ajustés.

	Apprentissage actif			Apprentissage supervisé	
	JM - *SLDC* -	wat4p1000 Reg. Log.	tftS2Fp1000 RO SVM	wat4pf Reg. Log.	tftS3Fpf RO SVM
1-AUC (%)	0.0093	0.0145	0.0144	0.0055	0.0093
TFP (%)	0.353	-	-	-	-
TFN (%)	0.173	-	-	-	-
Requêtes	943	< 1000	< 1000	75419	75419

Tab. 11.8: Comparaison de nos résultats avec ceux obtenus par Gordon Cormack (régression logistique) et D. Sculley (ROSVM) lors de TREC-07 [57], dans les tâches apprentissage actif et supervisé. Le corpus de messages de test est celui de TREC-07.

11.4.2 Le flot *JM-H/JM-S*

Nous avons simulé, pour le couple *JM-H/JM-S*, les différentes combinaisons des conditions de fonctionnement :

- marge d'apprentissage actif variant entre 0.05 et 0.50 (apprentissage supervisé - tous les messages sont utilisés pour l'apprentissage du classificateur) ;
- niveau de bruit dans la boucle de rétroaction variant entre 0 % (pas de bruit) et 16 % : l'information de classement fournie par le destinataire est fausse ;
- retard dans la boucle de rétroaction : entre 0 h (retour immédiat) et 168 h (1 semaine), avec des retard fixe ou aléatoire avec distribution exponentielle.

et évalué :

- l'erreur de classement spécifique par classe (faux positifs et faux négatifs) ;
- l'erreur de classement global selon 1-AUC % ;

- le nombre d'étiquettes demandées par le classificateur - pour l'apprentissage actif ;
- le nombre d'étiquettes retournées par le destinataire et effectivement prises en compte pour l'apprentissage ;
- la dimension finale du classificateur (nombre d'attributs).

Les résultats détaillés sont présentés sous différentes formes dans des tables dans l'Annexe G :

- les Tables G.5 et G.6 présentent les résultats en fonction du retard dans le retour d'information. Le retard est constant dans G.5 et aléatoire de distribution exponentielle (avec seuil de 10 minutes) dans G.6 ;
- les Tables G.7 à G.30 présentent les résultats en fonction de la marge d'apprentissage actif. Chaque table correspond à des valeurs fixes de retard dans le retour d'information d'étiquette et de niveau de bruit ;
- les Tables G.31 à G.38 synthétisent l'ensemble des Tables G.7 à G.30, ne présentant qu'une seule valeur : la dimension du classificateur construit (en nombre d'attributs) dans les Tables G.31 à G.34 et l'erreur de classement (1-AUC %) dans les Tables G.35 à G.38.

Une représentation synthétique intéressante est celle des Figures 11.5 et 11.6, où les *variables expliquées* - l'erreur de classement (*1-AUC%*), la dimension du classificateur et la fraction de messages classés utilisés pour l'apprentissage - sont représentées par des échelles de couleur en fonction de deux *variables explicatives* - la marge d'apprentissage actif et le niveau de bruit. Pour chaque variable expliquée, une série de quatre figures est utilisée pour représenter quatre valeurs de la troisième variable explicative : le retard dans la boucle de rétroaction.

En effet, le résultat le plus intéressante est l'apport de l'apprentissage actif par rapport à l'apprentissage supervisé. Pour mémoire, dans un contexte d'apprentissage supervisé la construction du classificateur utilise l'ensemble des exemples qui lui sont présentés, tandis dans celui d'apprentissage actif seuls ceux apportant une amélioration de la qualité de classement, selon un certain critère sont utilisés. L'*intensité* de l'apprentissage actif est contrôlé par la marge.

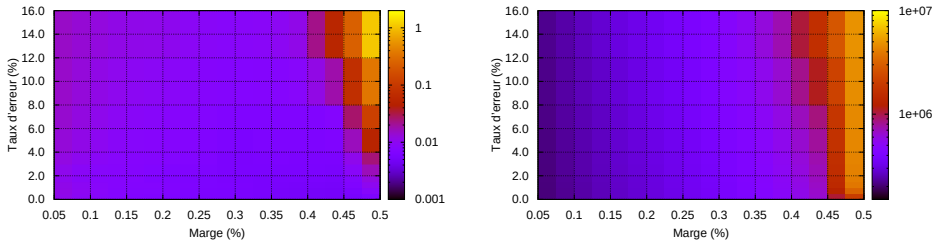
La Figure 11.5, colonne de gauche, montre l'erreur de classement (1-AUC %) en fonction de la marge et du niveau de bruit pour quatre niveaux de retard. On voit que, lorsqu'il n'y a pas de retard, l'erreur de classement croit avec les deux variables explicatives mais, en présence de retard la valeur optimale de marge prend une valeur intermédiaire, qui dépend du niveau de bruit et surement aussi des caractéristiques du flot et de l'algorithme de classement.

La valeur optimale de marge, au regard de l'erreur de classement, se trouve dans la région où la couleur bleue est la plus intense. L'existence d'une région optimale apparait plus clairement lorsqu'il y a retard dans la boucle de retour d'information. On peut interpréter cela par l'adéquation du classificateur au flot à filtrer : le classificateur est sur-ajusté pour des valeurs de marge plus importants et sous-ajusté pour des valeurs plus faibles.

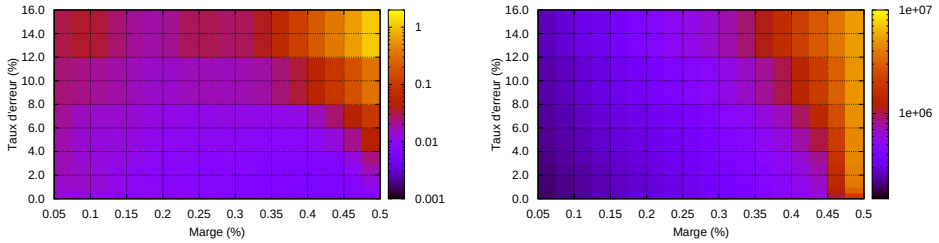
Dans la Table G.7, la valeur de la marge d'apprentissage actif qui correspond au point optimal de fonctionnement, du point de vue de l'erreur de classement (1-AUC %), semble se situer autour de 0.45. Il s'agit d'un contexte sans bruit et avec retour d'information immédiat. En dehors de ce contexte, la valeur de marge idéale est inférieure et se situe, la plupart du temps entre 0.30 et 0.35.

La colonne de droite de cette même Figure 11.5 montre la dimension finale du classificateur construit, qui croit très rapidement et en même temps que les variables explicatives.

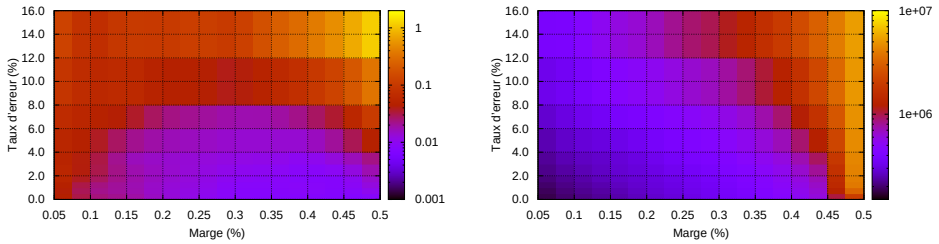
La Figure 11.6 présente, à gauche, la fraction de messages pour lesquels le classificateur demande le bon classement et, à droite, la fraction de messages effectivement utilisée pour l'apprentissage. On remarque que, au fur et à mesure que le retard augmente, l'écart entre ces deux indicateurs augmente aussi : le classificateur demande plus d'étiquettes qu'il n'utilise. Ceci est du au fait que, pendant l'attente de l'information de classement d'un message, des demandes concernant des messages similaires peuvent être faites, des demandes qui, même si satisfaites, ne seront plus pertinentes lors de leur retour.



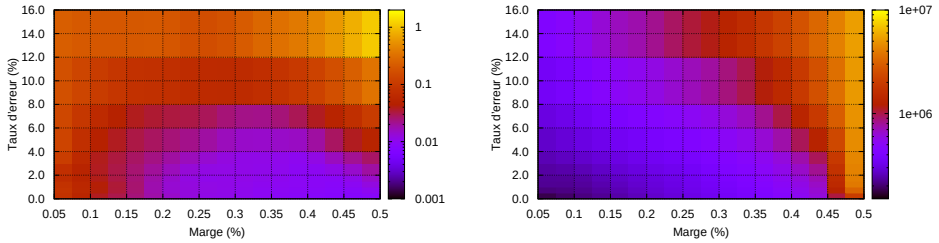
(a) Sans retard



(b) Retard exponentiel de moyenne 6 h

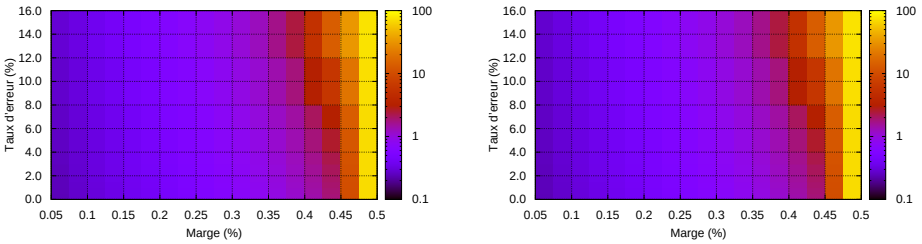


(c) Retard exponentiel de moyenne 24 h

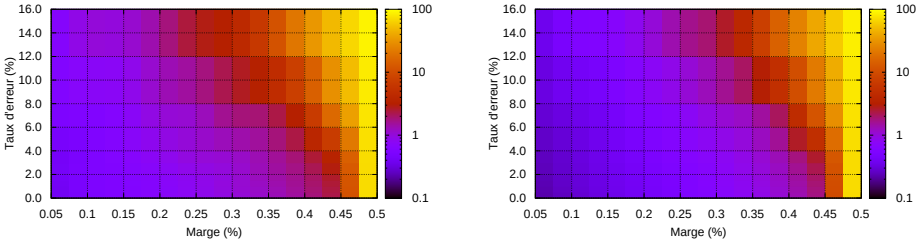


(d) Retard exponentiel de moyenne 48 h

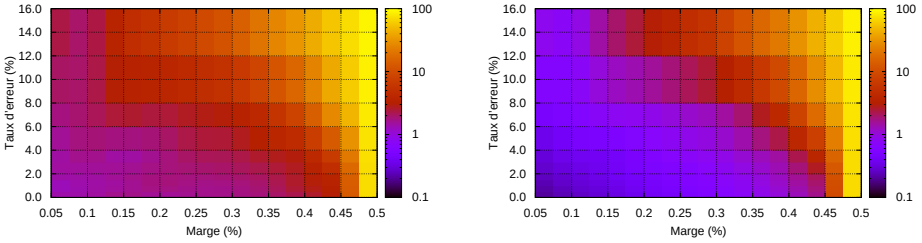
FIG. 11.5: Erreur de classement - 1-AUC (%) (figures de gauche) et dimension du classificateur construit - nombre d'attributs (figures de droite), en fonction de la marge d'apprentissage actif et du taux d'erreur dans le message de retour, pour différents niveaux de retard dans l'information de retour.



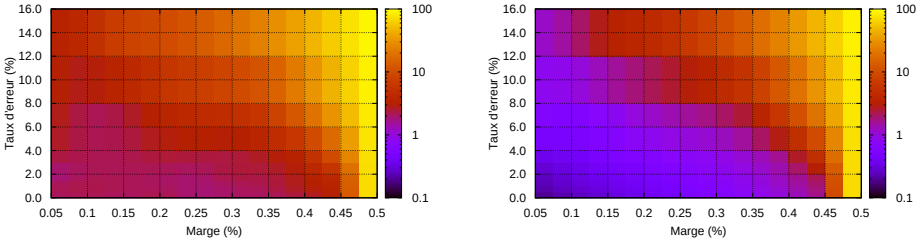
(a) Sans retard



(b) Retard exponentiel de moyenne 6 h

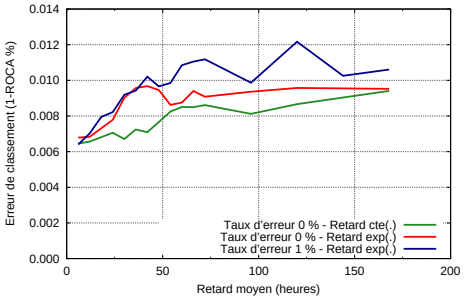


(c) Retard exponentiel de moyenne 24 h

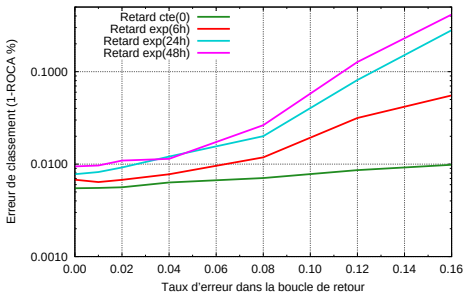


(d) Retard exponentiel de moyenne 48 h

FIG. 11.6: Dans les figures de gauche, le taux de requêtes (en % par rapport au nombre de messages traités), et à droit, le taux de messages effectivement utilisés pour l'apprentissage (en % par rapport au nombre de messages traités), en fonction de la marge d'apprentissage actif et du taux d'erreur dans le message de retour, pour différents niveaux de retard dans l'information de retour.

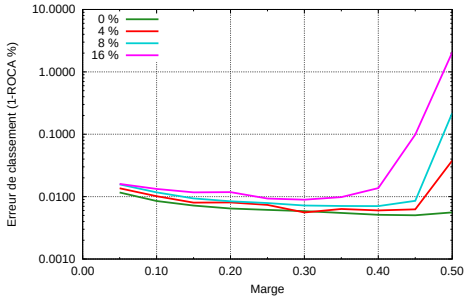


(a) 1-AUC% en fonction du délai de retour d'information, pour différents

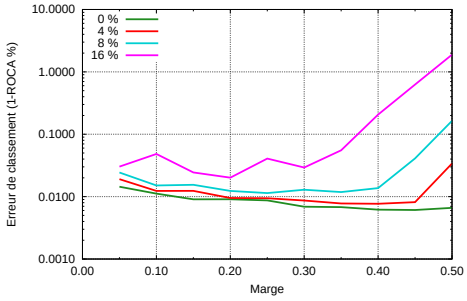


(b) 1-AUC% en fonction du niveau de bruit dans l'information de retour

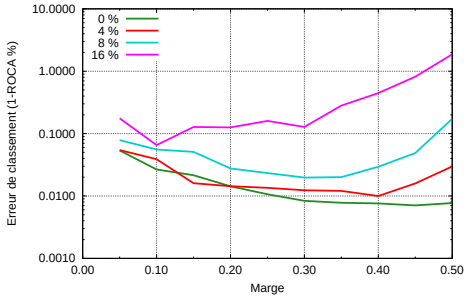
FIG. 11.7: La détérioration de l'information de retour en boucle fermée, soit par l'introduction d'un retard, soit par du bruit provoque une augmentation dans l'erreur de classement (1-AUC%). L'irrégularité remarqué dans la Figure 11.7a est due à l'interaction des délais avec les composantes périodiques du flot de messages.



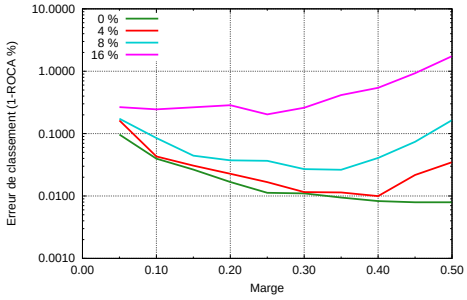
(a) Retour immédiat



(b) Retard moyen de 6 h



(c) Retard moyen de 24 h



(d) Retard moyen de 48 h

FIG. 11.8: Erreur de classement (1-AUC %) en fonction de la marge d'apprentissage actif, en fonction de la marge, dans différents environnements (niveau de bruit et retard de retour d'information).

11.5 Expérimentation de classement mutualisé

Utilisant le même contexte de l'expérimentation de la section précédente, nous avons ajouté les messages de quatre autres destinataires, un à la fois, puis tous ensemble. Les messages, légitimes, ont été collectés pendant une durée de deux mois (février et mars 2010), à l'exception de CB pour qui nous avons pu obtenir seulement les messages obtenus sur une durée d'un mois. Ces quatre destinataires ont des profils assez différents : enseignant/chercheur en finance quantitative, enseignant/chercheur en gestion de l'innovation, enseignant/chercheur en génie des procédés et informaticien de gestion.

Les résultats sont présentés dans les Tables 11.9 pour un marge d'apprentissage actif de 0.30 et 11.10 pour un marge de 0.35.

Ces résultats sont presque inattendus, sous plusieurs aspects.

Lorsque l'on mélange l'ensemble des utilisateurs l'efficacité mesurée par 1-AUC% est meilleure que lorsqu'on ajoute un seul utilisateur. La dégradation de l'erreur spécifique est plus importante dans la classe *spam* que dans la classe *ham*. De même que le taux de requêtes générées par l'apprentissage actif ne croit pratiquement pour la classe *ham*. La dimension du classificateur ne croit pas de façon significative.

Ces résultats ne peuvent surement pas être généralisés et méritent des investigations complémentaires : expérimentation dans un autre contexte du meme genre (institution enseignement/-recherche), mais avec un périmètre plus large (nombre d'utilisateurs plus important).

11.6 Conclusions

Les expérimentations faites dans ce chapitre constituent, à notre connaissance, la première expérimentation effectuée avec des ensembles de messages réels, collectés sur une durée assez longue. Néanmoins, il s'agit d'un contexte encore limité, par le faible nombre de destinataires ayant participé, mais surtout par une situation spécifique qui est la boite aux lettres d'un individu en France. Ces résultats appellent des nouvelles expérimentations de contenu similaire, mais dans d'autres contextes. Malgré ces restrictions, on peut penser que, qualitativement, les résultats dans d'autres contextes ne doivent pas être trop différents.

Ainsi, utilisant des ensembles synthétiques de messages, à partir de listes de discussion et de pièges à spam, nous avons obtenu des résultats presque parfaits. Les résultats obtenus avec des ensembles réels de messages ont été moins bons, mais avec des taux d'erreur qui restent encore bien inférieurs à 1 %.

Le fonctionnement en apprentissage actif en ligne, grâce à une boucle de rétroaction, a permis de maintenir ce niveau d'efficacité tout au long de l'expérimentation, avec une faible contrainte pour le destinataire des messages, puisque moins de 1 % des messages classés ont fait objet d'une demande d'information de bon classement (apprentissage actif).

À l'ensemble initial de messages qui ne concernait un seul destinataire, nous avons ajouté, pendant deux mois, les messages de quatre autres destinataires avec des profils différents du premier e nous avons constaté que la perte en efficacité marginale n'a pas été très significative.

L'algorithme de classement utilisé est assez simple et linéaire, donc incapable de séparer des classes dont la frontière est autre chose qu'un hyperplan. L'apprentissage en ligne est basé sur de l'approximation stochastique et l'apprentissage actif utilise une marge fixe.

On peut donc penser que le problème de classement de messages électroniques n'est pas un problème complexe. En fait, ce qui le rend difficile est l'intolérance des destinataires : la recherche de la perfection.

Il reste à savoir que faut-il faire pour obtenir de meilleurs résultats. Pour cela, il y a deux raisonnements :

- soit les classes sont séparables et il faudrait un classificateur non linéaire et plus complexe ;
- soit les classes sont non séparables à cause d'objets dont l'étiquette correcte dépend d'informations qui ne peuvent pas être codées de façon déterministe dans le modèle de repré-

Corpus	**Classe**	**Messages**		**Erreurs**			**1-ROCA%**		**Query**		**Dim.**
		Nb		Nb		%	%	Intervalle de Confiance	Nb	%	Attributs
JM-H	ham	10025		10		9.975e-02	3.991e-03	(7.160e-04 - 2.224e-02)	106	1.057	326601
JM-S	spam	27896		10		3.585e-02			117	4.194e-01	
JM-H + JMV	ham	10599	(574)	19	(10)	1.793e-01	3.192e-03	(1.107e-03 - 9.203e-03)	129	1.217	330971
JM-S	spam	27896		14		5.019e-02			149	5.341e-01	
JM-H + AG	ham	11419	(1394)	18	(7)	1.576e-01	3.239e-03	(9.195e-04 - 1.141e-02)	143	1.252	334631
JM-S	spam	27896		14		5.019e-02			161	5.771e-01	
JM-H + CB	ham	10293	(268)	15	(7)	1.457e-01	5.145e-03	(1.235e-03 - 2.143e-02)	132	1.282	333503
JM-S	spam	27896		13		4.660e-02			149	5.341e-01	
JM-H + TW	ham	13054	(3029)	19	(10)	1.455e-01	5.159e-03	(1.323e-03 - 2.011e-02)	145	1.111	334873
JM-S	spam	27896		14		5.019e-02			165	5.915e-01	
JM-H + TOUS	ham	15290	(5265)	20	(0,2,0,7)	1.308e-01	3.742e-03	(9.545e-04 - 1.467e-02)	159	1.040	337048
JM-S	spam	27896		13		4.660e-02			177	6.345e-01	

TAB. 11.9: Synthèse des résultats de classement avec apprentissage actif et en ligne, dans des conditions idéales - rétroaction non bruitée et sans retard. Ces résultats sont à comparer avec ceux obtenus avec un classificateur en boucle ouverte (Table 10.1). m=0.30

Corpus	**Classe**	**Messages**		**Erreurs**			**1-ROCA%**		**Query**		**Dim.**
		Nb		Nb		%	%	Intervalle de Confiance	Nb	%	Attributs
JM-H	ham	10025		13		1.297e-01	4.068e-03	(7.436e-04 - 2.225e-02)	141	1.406	374475
JM-S	spam	27896		10		3.585e-02			160	5.736e-01	
JM-H + JMV	ham	10599	(574)	21	(9)	1.981e-01	3.639e-03	(1.322e-03 - 1.002e-02)	168	1.585	379425
JM-S	spam	27896		15		5.377e-02			206	7.385e-01	
JM-H + AG	ham	11419	(1394)	21	(10)	1.839e-01	3.290e-03	(1.261e-03 - 8.584e-03)	193	1.690	385557
JM-S	spam	27896		15		5.377e-02			225	8.066e-01	
JM-H + CB	ham	10293	(268)	17	(7)	1.652e-01	6.827e-03	(1.795e-03 - 2.596e-02)	170	1.652	381421
JM-S	spam	27896		14		5.019e-02			200	7.169e-01	
JM-H + TW	ham	13054	(3029)	22	(11)	1.685e-01	4.392e-03	(1.837e-03 - 1.050e-02)	199	1.524	387094
JM-S	spam	27896		15		5.377e-02			231	8.281e-01	
JM-H + TOUS	ham	15290	(5265)	20	(0,2,0,8)	1.308e-01	3.477e-03	(9.393e-04 - 1.287e-02)	210	1.373	387703
JM-S	spam	27896		15		5.377e-02			239	8.568e-01	

TAB. 11.10: Synthèse des résultats de classement avec apprentissage actif et en ligne, dans des conditions idéales - rétroaction non bruitée et sans retard. Ces résultats sont à comparer avec ceux obtenus avec un classificateur en boucle ouverte (Table 10.1). m=0.35

sentation des messages. Par exemple, des messages dont le classement correct dépend du destinataire ou du moment.

Il reste que l'ordre de grandeur de l'erreur de classement est déjà égal ou inférieur à l'imprécision des différents paramètres attachés aux causes de l'erreur de classement.

Par exemple, les événements périodiques détectés dans le chapitre précédent ne semblent pas avoir impacté significativement les résultats dans le contexte d'apprentissage actif en ligne. Néanmoins, à cause des interactions entre ces événements et les délais de retour d'information, on peut penser que ces événements pourraient finir par avoir un impact de plus en plus important, au fur et à mesure que l'on atteint des taux d'erreur de classement de plus en plus faibles.

Enfin, nous avons comparé les résultats de classement de la méthode que nous proposons ici avec ceux qui ont présenté les meilleurs résultats lors de TREC 2007 - SVM et une proposition de régression logistique - et nos résultats ont été légèrement meilleurs.

Dans le classificateur (algorithme plus contexte d'apprentissage) que nous proposons, deux paramètres méritent une étude plus approfondie pour leur rendre adaptatifs :

- la marge d'apprentissage actif pourrait évoluer dans le temps en fonction de la valeur des certains paramètres évalués en temps réel, *e.g.* le niveau de bruit ou le retard dans le retour d'information ;
- la vitesse d'apprentissage pourrait dépendre aussi des paramètres de fonctionnement évalués en temps réel au lieu de dépendre uniquement du temps écoulé depuis l'initialisation du système, de façon à s'adapter plus rapidement aux changements brusques dans le flot.

L'apprentissage actif s'est montré particulièrement intéressant comme moyen pour assurer le meilleur ajustement du classificateur : ni en excès ni en manque. Pour que cela soit possible, il faut que la marge puisse devenir un paramètre adaptatif et non pas statique comme il a été le cas ici.

L'apprentissage actif en ligne présente aussi un intérêt additionnel assez important : en effet, dans une application de classement de messages électroniques habituel, il est pratiquement impossible d'évaluer l'efficacité de filtrage. On peut penser que les informations retournées par le destinataire pourraient permettre d'évaluer, en ligne, des paramètres tels l'efficacité de classement et la distribution des retards dans la boucle de retour.

Cinquième partie

Réflexions à approfondir

CHAPITRE 12

Geométrie des classes et filtrage parfait

> Far better an approximate answer to the right question, than the exact answer to the wrong question, which can always be made precise.
>
> *John Tukey*

12.1 Introduction

Classement sans erreurs ? ? ? Dans la communauté d'apprentissage artificiel, il s'agit d'une question qui n'a pas vraiment de sens et que ne se pose donc pas (ou plus).

Ceux qui utilisent des filtres anti-spam cherchent le "filtrage parfait" le croyant exister. Les développeurs de logiciels de filtrage annoncent le filtre (presque) parfait [269]. Des tests comparatifs de produits commerciaux (*e.g.* [119]) présentent des résultats aussi (presque) parfaits. Le filtre parfait, c'est le "Graal" des practiciens ! ! ! Un examen plus attentif de ces résultats montre que le contexte d'expérimentation n'est pas toujours réaliste.

La question de la "perfection" d'un filtre anti-spam n'est pas spécifique au thème de cette thèse, mais nous avons estimé utile de l'étudier et d'apporter une réponse satisfaisante.

Il n'est pas question de pouvoir quantifier la "perfection" d'un classificateur, mais on peut se contenter de pouvoir expliquer pourquoi le filtrage peut, dans le meilleur des cas, être "optimal" et jamais "parfait".

Il y a, bien évidemment, des raisons liées à l'algorithme de classement : certains peuvent être plus efficaces que d'autres. Mais il y a aussi des raisons liées aux données. C'est sur cet aspect que nous développons d'abord l'idée de *séparabilité* : condition nécessaire pour qu'une hypothèse puisse correspondre parfaitement aux objets à classer. Ensuite nous explorons quelques raisons structurelles des données (distribution spatiale) qui font que la généralisation est meilleure dans certaines situations plutôt que d'autres. Dans cette partie nous analysons le problème de classement avec un point de vue d'apprentissage de fonctions booléennes.

12.2 Séparabilité et Classement sans Erreur

Nous avons, dans la Section 5.3.1, présenté la séparabilité des classes comme un pré-requis essentiel pour le classement sans erreur, *c.à.d.* le filtrage parfait. Lorsque les classes ne sont pas séparables, aucun classificateur n'est capable de classer les éléments de l'ensemble sans erreur.

Parmi les causes de non séparabilité, on peut citer l'étiquetage non déterministe, *e.g.* des exemples qui ne sont pas appréciés de la meme façon par des destinataires différents. Ceci pourrait être interprété comme une représentation de dimension insuffisante, puisqu'il suffirait d'ajouter une dimension - le destinataire, avec les préférences, mais une telle solution n'est pas toujours réaliste et il convient parfois de considérer que l'étiquette de certains exemples est une donnée aléatoire.

La séparabilité des classes assure l'existence d'au moins une hypothèse capable de classer tout élément, sans erreur. Néanmoins, cela ne suffit pas. Selon la structure de l'ensemble à classer (que ce soit celui des objets bruts ou des représentations), il est possible que cet objectif ne soit atteignable qu'après un apprentissage exhaustif sur tous les objets et le classificateur perd tout son intérêt puisqu'il n'a aucune capacité de généralisation, à cause de la structure des classes et non pas de l'algorithme de classement ou d'apprentissage.

La notion de séparabilité linéaire est aussi intéressante puisqu'elle correspond à des algorithmes de classement moins complexes.

Zighed [276] suggère que la capacité d'apprentissage d'une méthode est fortement associée au niveau de séparabilité des classes et que *les classes sont plus facilement séparables* si :

1. les instances de la même classe apparaissent plutôt regroupées dans dans le même groupe dans l'espace de représentation ;
2. le nombre de groupes est faible, pas beaucoup plus important que le nombre de classes ;
3. les frontières entre les groupes ne sont pas complexes

Il conviendrait d'ajouter *"les hypothèses générées sont plus simples ou la généralisation est plus aisée"* à la phrase *"les classes sont plus facilement séparables"*.

Thornton [250] étudie la séparabilité géométrique, en opposition à la séparabilité linéaire, e propose *GSI* (*Geometric Separability Index*) un indicateur de séparabilité permettant d'évaluer la séparabilité des classes, y compris des classes dont la représentation spatiale contient des régions non connexes. Ses travaux portent principalement sur le classificateur *C4.5* (des arbres de décision) : il serait intéressant de reprendre sa proposition et l'appliquer à d'autres classificateurs.

Dans la section suivante nous étudions quelques aspects de la distribution spatiale des exemples qui tentent de justifier cette suggestion pour ensuite essayer de comprendre le cas spécifique du classement de messages électroniques.

12.3 Distribution spatiale des données

12.3.1 Apprentissage de fonctions booléennes

Il est assez courant qu'un processus d'apprentissage soit modélisé comme l'apprentissage d'une fonction booléenne (*e.g.* dans le modèle PAC d'apprentissage [142]). Cette représentation convient bien à une application de classement de messages électroniques utilisant un modèle du type Bag of Words, puisque les messages sont représentés par un vecteur d'attributs (les termes d'un dictionnaire) et dont la valeur associée à chaque attribut est la présence ou absence dans le message[1]. La dimension du vecteur représentant le message est la taille du dictionnaire :

$$M = (x_1, x_2, \ldots, x_{|V|}),\ x_i \in \{0, 1\} \tag{12.1}$$

L'ensemble des tous les messages que l'on peut construire avec ce vocabulaire ($2^{|V|}$, si l'on considère que toutes les combinaisons sont possibles), constitue un treillis booléen[2]. Les Bornes Supérieure et Inférieure de cet ensemble sont les vecteurs de dimension $|V|$: $I = (1, 1, \ldots, 1)$ et $O = (0, 0, \ldots, 0)$.

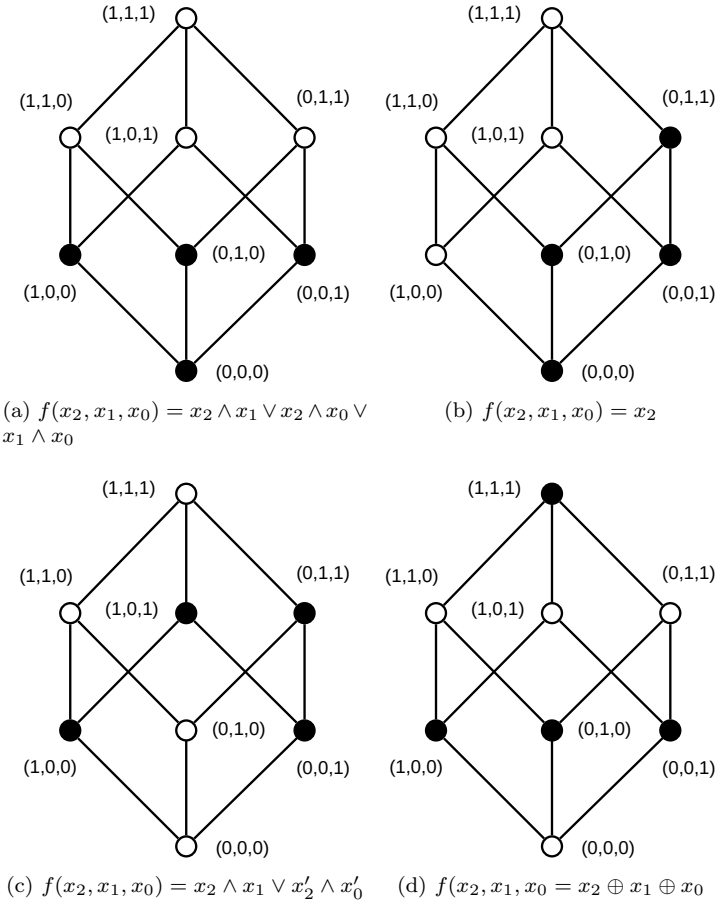
[1] Si l'on utilise le nombre d'apparitions de chaque terme, au lieu de sa présence, on peut utiliser $\lceil \log n \rceil$ attributs binaires, n étant le nombre de valeurs différents à codifier

[2] Pour une introduction aux treillis booléens, consulter l'annexe F.

La dimension des vocabulaires que l'on trouve dans les applications de filtrage de spam varie entre quelques dizaines de milliers à quelques millions d'attributs. La figure (12.1) montre un exemple simple avec seulement trois attributs (que l'on peut visualiser en trois dimension). La position relative des objets fait que les classificateurs soient plus ou moins complexes.

Dans la figure (12.1a) les éléments des deux classes sont regroupées en deux clusters. Tous les attributs sont pertinents et les classes sont linéairement séparables par l'hyperplan :

$$x_2 + x_1 + x_0 = 3/2$$



(a) $f(x_2, x_1, x_0) = x_2 \wedge x_1 \vee x_2 \wedge x_0 \vee x_1 \wedge x_0$ (b) $f(x_2, x_1, x_0) = x_2$

(c) $f(x_2, x_1, x_0) = x_2 \wedge x_1 \vee x'_2 \wedge x'_0$ (d) $f(x_2, x_1, x_0 = x_2 \oplus x_1 \oplus x_0$

Fig. 12.1: Diagramme de Hesse de configurations différentes des objets de classement : deux classes avec 4 éléments chacune. Les classes d'appartenance sont représentées par la couleur des sommets. (12.1a) et (12.1b) sont linéairement séparables. Dans (12.1b), un seul attribut est pertinent alors que tous les sont dans (12.1a). Deux hyperplans sont nécessaires pour séparer les classes en (12.1c). Dans le cas de la fonction *ou exclusif* (12.1d) aucun des sommets a un voisin dans la même classe et il faut, au moins, 3 hyperplans pour séparer les classes.

Les objets de la figure (12.1a) sont aussi regroupés en deux clusters, mais on voit qu'un seul attribut suffit à séparer les classes. C'est le cas ou des parties des vocabulaires des classes sont disjoints (p. ex., des langues différentes). Les classes sont linéairement séparables par l'hyperplan :

$$x_2 = 1/2$$

Dans l'exemple de la figure (12.1c), les objets sont disposés sur des chaînes et, même s'ils constituent toujours deux clusters, ils ne sont pas linéairement séparables par une seule hypothèse

et il faut deux hyperplans pour séparer les classes (la solution n'est pas unique) :

$$x_2 + x_0 = 1/2$$
$$x_2 + x_1 = 3/2$$

Enfin, la figure (12.1c) représente la fonction *ou exclusif*, où aucun élément n'a un voisin de la même classe. Il n'y a pas d'hypothèse linéaire comportant un hyperplan unique capable de séparer les classes et il en faut trois :

$$x_2 + x_1 + x_0 = 1/2$$
$$x_2 + x_1 + x_0 = 3/2$$
$$x_2 + x_1 + x_0 = 5/2$$

Ces exemples, dans un espace de dimension encore très faible, montrent l'influence de la disposition spatiale relative des objets sur la complexité de l'hypothèse capable de les séparer. Ces interactions apparaissent naturellement dans des dimensions plus importantes.

La première remarque que l'on peut faire de l'observation de ces exemples est que la situation où la distribution spatiale des éléments est la plus uniforme (figure 12.1d) est celle où l'hypothèse est la plus complexe. Idéalement, les situations où les classes sont linéairement séparables par un seul hyperplan sont les plus intéressantes, puisqu'elles peuvent être traitées plus facilement par des algorithmes de classement linéaires courants (Bayes Naïf, Régression Logistique, Machine à Vecteur de Support, ou l'algorithme discriminant linéaire utilisé dans nos travaux).

On observe aussi que la distance entre chaque élément et la borne inférieure du treillis correspond au nombre d'attributs actifs. Les éléments avec un nombre identique d'attributs actifs se retrouvent regroupés au même niveau (voir figure 12.2).

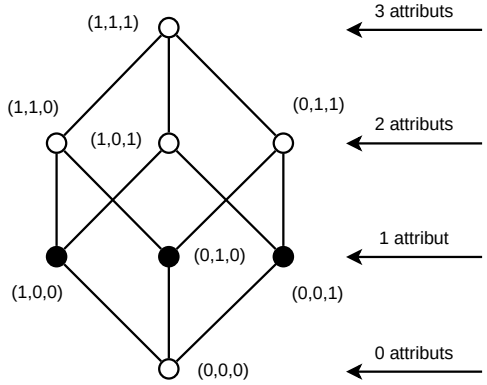


FIG. 12.2: Dans une représentation des classes utilisant des treillis, tous les objets se trouvant à un même niveau ont le même nombre d'attributs.

Un vocabulaire de dimension $|V|$ génère un treillis avec $2^{|V|}$ sommets : tous les messages possibles pouvant avoir jusque à $2^{|V|}$ termes distincts. Mais nous sommes plutôt intéressés par les messages avec une diversité bien plus faible. Les pratiques courantes de sélection d'un nombre fixe d'attributs ou de normalisation de la longueur des documents fait que les représentations utiles se trouvent toujours à une distance fixe (ou presque) de la base du treillis. En fait, seuls les messages dont le nombre de termes distincts est bien inférieur à la dimension du vocabulaire ($\ell << |V|$), intéressent les applications de classement. Pour un nombre donné ℓ d'attributs distincts, le nombre de combinaisons possibles est :

$$N(\ell) = \binom{|V|}{\ell} = \frac{|V|!}{\ell!\ (|V|-\ell)!} = \prod_{k=1}^{\ell} \frac{|V|-\ell+k}{k} \tag{12.2}$$

12.3.2 Discussion

La table 12.1 montre des exemples d'ordres de grandeur du nombre de termes utiles que l'on peut trouver dans des classificateurs (évalués selon 12.2). Ces valeurs sont juste indicatives, et il faudrait encore retrancher les combinaisons qui ne peuvent pas exister dans une langue.

Représentation	$\|V\|$	# sommets	ℓ	# sommets utiles
Mots et bi-mots (Bayes Naïf)	1000000	$2^{1000000}$	64	$\approx 2^{980}$
4-grams (*SLDC*)	1000000	$2^{1000000}$	3500	$\approx 2^{33587}$

Tab. 12.1: Exemples d'ordres de grandeur du nombre de sommets potentiels et utiles pour deux classificateurs : Bayésien Naïf et le classificateur discriminant linéaire utilisé dans cette thèse. Le premier sélectionne les ℓ attributs les plus discriminants tandis que le deuxième utilise une longueur fixe du message. Ces valeurs ont été calculées à l'aide de (12.2)

L'information à retenir de ces chiffres n'est pas tant les valeurs, qui peuvent être différentes dans d'autres situations, mais plutôt que le nombre de sommets effectivement utiles est négligeable par rapport au nombre de sommets du treillis et se trouvent concentrés dans une région à une distance plus ou moins constante de la base du treillis.

A noter aussi que cette distribution ne correspond pas à une hypothèse usuellement considérée dans des modèles tels PAC [142] où les exemples seraient uniformément distribués dans le espace de représentations.

Si cette distribution spatiale dans un treillis rend, probablement, moins évidente la séparabilité linéaire des classes, les sommets non utilisées peuvent être attribués indifféremment à l'une ou l'autre classe de façon privilégier les hypothèses plus simples.

Le cas trivial de séparation linéaire aisée est celui des classes avec des vocabulaires disjoints (l'équivalent de la figure 12.1b), où l'intersection des vocabulaires des classes est vide. Dans la section suivante nous présentons quelques résultats expérimentaux.

La représentation graphique de la distribution spatiale permet aussi d'établir un lien entre cet aspect et la capacité de généralisation, en tant que propriété intrinsèque des données. Dans une application d'apprentissage il est important de pouvoir généraliser des conclusions à l'ensemble des cas possibles, à partir de l'observation d'un nombre limité d'exemples et bien sur non exhaustif. Le cas extrême est la fonction *ou-exclusif* où une hypothèse consistante ne peut-être construite qu'après observation exhaustive de tous les cas possibles, donc, aucune possibilité de généralisation.

12.4 Vocabulaires des classes : communs ou disjoints ?

Nous avons vu dans la section précédente, à l'aide de la représentation spatiale des objets, que les attributs spécifiques à chaque classe facilitent le classement par des hypothèses plus simples.

Ce constat peut être fait pour chaque algorithme de classement. Par exemple, dans le cas du classificateur bayésien naïf utilisant le modèle multinomial avec des attributs booléens la règle de

décision résulte de la comparaison des vraisemblances pondérées par la probabilité à priori des classes (voir équations (6.2) et (6.9)) :

$$\begin{aligned} \hat{y} &= \underset{c \in \{ham,spam\}}{\arg\max} \; P(c)\, P(M \mid c) \\ &= \underset{c \in \{ham,spam\}}{\arg\max} \; P(c) \prod_{t \in \mathcal{V}} P(t \mid c)^{\mathbb{I}_p(t,\mathbf{m})} \end{aligned} \tag{12.3}$$

Si un message contient des termes qui n'existent que dans une des classes, la vraisemblance sera nulle pour l'autre classe. Cela fait que la valeur exacte des probabilités des termes partagés ainsi que la probabilité à priori des classes a très peu d'importance.

Mais cela est intuitivement évident. L'extension extrême serait que chaque classe ait son propre vocabulaire. Cela est évident intuitivement.

Dans cette section, nous présentons quelques résultats de la comparaison des vocabulaires de hams et spams, tels que vus par deux filtres. Ce que nous intéresse plus particulièrement est d'évaluer leur degré de superposition, *c.à.d.* quelle fraction des termes d'un corpus est partagée par deux classes et quelle fraction est constituée uniquement de termes qui n'apparaissent que dans une seule classe.

Nous avons étudié les vocabulaires obtenus lors de l'apprentissage de deux classificateurs sur 7 corpus différents (Table 12.2). Nous avons relevé, pour chaque corpus et filtre, le nombre total d'attributs et retenu seulement ceux vus plus d'une fois dans les messages (les non *hapax*) et comptabilisé les attributs exclusifs à une classe, leur fraction dans la classe et la fraction d'attributs exclusifs dans le vocabulaire entier.

Corpus	Définition	Hams	Spams
JM-1003	Corpus de JM du mois de mars 2010	5007	13746
TW-1003	Corpus de TW du mois de mars 2010	3455	897
JM+TW-1003	mélange de JM-1003 et TW-1003	8482	14643
JM-1003-EN	JM-1003 - classe HAM avec les seuls messages en anglais	3090	13746
JM-1003-FR	JM-1003 - classe HAM avec les seuls messages en français	1862	13746
JM-09xx	Corpus de JM de l'année 2009	51444	261759
TREC-07	Corpus de TREC 2007 Spam Track	25220	50199

TAB. 12.2: Corpus utilisés pour étudier la superposition des vocabulaires des spams et des hams

Les résultats d'un filtre *bayésien naïf* sont présentés dans la table Tab. (12.3). Ce filtre utilise les *mots* et les *bi-mots* comme attributs. Les attributs sont extraits du corps du message et les seuls en-têtes utilisés sont le *Sujet* et l'*Expéditeur*. L'apprentissage se fait hors-ligne (batch).

Le deuxième classificateur (*SLDC*), est basé sur un algorithme discriminant linéaire simple et utilise des *4-grams* comme attributs. Les résultats sont présentés dans les tables Tab. (12.4) pour l'apprentissage passif (SLDC-AP) et (12.5) pour l'apprentissage actif (SLDC-AA). Au contraire du filtre bayésien naïf, ce filtre utilise les en-têtes et une longueur fixe du corps des messages (512 caractères). L'apprentissage se fait en ligne par approximation stochastique.

Les valeurs numériques obtenues sont dans des plages différentes selon l'algorithme de classement et la représentation utilisée mais ils reflètent, à l'intérieur d'une même table, la difficulté de classement intuitive : des chiffres plus faibles indiquent plus de termes partagés par les deux classes.

Le corpus JM-1003 et TW-1003 sont constitués des spams et des messages légitimes reçus par JM (informaticien) et TW (chercheur en économie) pendant le mois de mars 2010. Les messages de TW sont moins prévisibles que celles de JM et, à priori, plus difficiles à classer. C'est ce qui

Bayes Naïf	**Attributs**		**Spam**		**Ham**		**Total**
Corpus	total	s/ hapaxes	excl	% excl	excl	% excl	% excl
JM-1003	2042560	580963	313683	86.25	217292	81.30	91.40
TW-1003	677916	288665	86884	76.08	174461	86.46	90.54
JM+TW-1003	2601578	820094	375859	82.97	367088	82.63	90.59
JM-1003-FR	1686399	440338	344730	94.80	76709	80.23	95.71
JM-1003-EN	1860898	503606	325128	89.41	139977	78.43	92.35
JM-09xx	11162512	3150887	1501553	88.32	1450789	87.96	93.70
TREC-07	4740152	1514408	661756	82.97	716850	84.07	91.03

Tab. 12.3: Vocabulaires extraits par un filtre Bayésien Naïf (apprentissage hors-ligne) et attributs exclusifs à une seule classe. Les attributs sont des mots et bi-mots

SLDC - A. Passif	**Attributs**		**Spam**		**Ham**		**Total**
Corpus	total	s/ hapaxes	excl	% excl	excl	% excl	% excl
JM-1003	1113613	382370	159942	50.07	62946	28.30	58.29
TW-1003	366736	190687	22867	25.44	100791	60.06	64.85
JM+TW-1003	1283562	486576	156907	41.93	112402	34.10	55.35
JM-1003-FR	918754	308842	196905	68.58	21743	19.42	70.80
JM-1003-EN	1029631	342957	173448	58.01	43962	25.93	63.39
JM-09xx	1639588	519902	205452	47.94	91362	29.05	57.09
TREC-07	1884213	466183	168191	42.39	69420	23.30	50.97

Tab. 12.4: Vocabulaires extraits par *SLDC*, le classificateur discriminant linéaire simple utilisé dans cette thèse (apprentissage passif en ligne, par approximation stochastique), et attributs exclusifs à une seule classe. Les attributs sont des 4-grams de niveau caractère

SLDC - App. Actif	**Attributs**		**Spam**		**Ham**		**Total**
Corpus	total	s/ hapaxes	excl	% excl	excl	% excl	% excl
JM-1003	201802	75713	24573	39.91	14144	27.66	51.14
TW-1003	163783	71732	12360	24.35	20973	35.32	46.47
JM+TW-1003	299522	122143	28068	29.64	27438	29.17	45.44
JM-1003-FR	142717	53978	21165	46.84	8792	26.79	55.50
JM-1003-EN	151935	55204	21084	45.54	8903	26.09	54.32
JM-09xx	239008	94213	31841	41.59	17659	28.31	52.54
TREC-07	174401	72787	22936	38.27	12853	25.78	49.17

Tab. 12.5: Vocabulaires extraits par *SLDC*, le classificateur discriminant linéaire simple utilisé dans cette thèse (apprentissage actif en ligne, par approximation stochastique) et attributs exclusifs à une seule classe. Les attributs sont des 4-grams de niveau caractère

ressort des résultats, sauf pour le classificateur SLDC-AP. Nous attribuons cela au nombre réduit de messages dans le corpus TW-1003 et au fait que l'apprentissage passif utilise tous les messages disponibles sans chercher à optimiser la qualité de classement.

Sachant que la langue dominante dans les spams est l'anglais, nous avons expérimenté une extraction du corpus JM-1003 avec seulement les messages en français (JM-1003-FR) ou en anglais (JM-1003-EN). On remarque que la superposition est plus faible dans dans JM-1003-FR, mais pas aussi faible que l'on pourrait attendre. En fait, les méta-données des messages (formatage HTML et en-têtes) introduit une "troisième" langue, commune aux deux classes, qui diminue l'influence de la langue dominante dans les messages. Par ailleurs, nous avons vu, dans le Chapitre 10, que les informations extraites dans les méta-données semblent etre plus discriminantes que le contenu effectif du message.

Lorsque nous avons mélangé les boites aux lettres (JM-1003 + TW-1003) la superposition des vocabulaires a été plus faible, à cause de l'augmentation de la diversité des messages de la classe *ham*. Ceci suggère que le classement peut être plus difficile dans les situations d'utilisation partagée d'un classificateur de messages électroniques.

La comparaison des résultats de JM-09xx (messages reçus pendant une année entière) avec ceux de JM-1003, montre la stabilité des résultats pendant une période plus longue.

Enfin, le corpus TREC-07 permet de vérifier que l'on obtient des résultats similaires avec un corpus public, certes plus ancien, et dont la langue dominante dans les messages légitimes est la langue anglaise.

Le constat le plus intéressante de cet expérimentation est que dans les cas examinés - deux classificateurs différents, des messages d'origines très différentes avec des représentations aussi différentes - moins de la moitié des termes est partagée par les deux classes. Ce critère n'est pas le seul pris en compte pour le classement des messages, mais il explique en partie la facilité avec laquelle on obtient, avec peu d'effort et avec des classificateurs linéaires, des taux d'erreur aussi faibles (inférieurs à 5 %) dans le classement de messages électroniques.

Dans la Section 7.4, nous avons mentionné les méthodes de constitution des ensembles d'exemples par les filtres *j-chkmail* et *CanIt*. Le nombre important de termes exclusifs à chaque classe explique pourquoi les méthodes empiriques de ces filtres donnent des résultats aussi satisfaisants.

Cela explique aussi pourquoi certains *spams* utilisant un vocabulaire ressemblant fortement à celui de la classe *ham* - les *phishings* et certaines arnaques - sont plus difficiles à détecter.

Néanmoins, n'oublions pas que l'environnement de filtrage de spam est celui d'une "course aux armements" et que ces hypothèses peuvent évoluer dans le temps.

12.5 Discussion et Conclusions

La séparabilité des classes est une condition nécessaire pour que deux ensembles puissent être sépares sans erreur. La distribution spatiale des données défini la complexité du classificateur et la capacité de généralisation intrinsèque des données.

Les expérimentations ont démontré la facilité que l'on a pour séparer les *hams* des *spams*. On peut surement affirmer qu'il ne s'agit pas d'un problème difficile dans le sens où l'on peut atteindre des niveaux d'efficacité assez importants avec des classificateurs peu sophistiqués - des classificateurs linéaires. Néanmoins, rien ne dit que le classement sans erreur soit possible. Cela dépend de plusieurs autres aspects. Voici quelques raisons qui font que le taux d'erreur n'est pas nul ou même pas optimal :

– *Séparabilité vue à des niveaux différents d'interprétation* - La définition 5.1 est intentionnellement vague sur le type des éléments de l'ensemble. Normalement la bibliographie concernant l'apprentissage artificiel fait référence plutôt à la représentation (souvent vectorielle) des objets tel que vus par les algorithmes de classement (*e.g.* [256, p.401] [126, p. 121]). Les êtres humains observent les objets bruts et l'interprètent à un niveau sémantique alors que applications de filtrage observent leurs représentations vectorielles (avec perte d'information). La séparabilité des classes n'est pas la même selon le niveau d'analyse.

- *Apprentissage sur des exemples incorrectement classés* - Il a été remarqué que les utilisateurs commettent des erreurs lorsqu'on leur demande de classer un lot de messages, des erreurs pouvant atteindre 3%-7% [249] [118], alors que la précision d'un filtre anti-spam peut atteindre des taux d'erreur aussi faibles que 0.5 %. Ces erreurs, modélisées comme du bruit, impactent l'efficacité des filtres. Cormack, Kolcz et Sculley [62] [228] ont étudie le problème du point de vue spécifique au filtrage de spam, proposent des solutions, plus ou moins complexes, pour atténuer l'influence du bruit sans toutefois le supprimer complètement.
- *Éléments avec classement non déterministe (Gray Mail)* - Il s'agit d'une catégorie de messages qui ne sont pas catégorisés de façon identique selon le destinataire ou la situation du moment [249] [261]. Parfois il suffirait d'ajouter un attribut indicateur de la situation particulière. Cela n'est pas toujours possible et il conviendrait de considérer que le classement de tels cas relèvent d'un phénomène aléatoire dont la conséquence est de rendre les classes non séparables.
- *Apprentissage empoisonné* - Il s'agit du bruit éventuellement ajouté intentionnellement par du *spam*. Le filtrage de spam est une application fonctionnant dans un environnement hostile. Un des nombreuses méthodes pour attaquer les filtres serait, par exemple, rejouer des messages légitimes tout en changeant juste un lien d'un site web [179] [180].
- *Algorithme de classement linéaire pour un problème non linéaire* - Malgré les résultats très bons que l'on peut obtenir avec des classificateurs linéaires, rien ne prouve que les classes impliquées soient séparables et si oui qu'elles soient séparables par un classificateur linéaire.
- *Mauvaise utilisation du filtre* - Malgré les excellents résultats théoriques que l'on peut obtenir dans les applications de classement de spams, une gestion déficiente de l'application de filtrage (généralement liée à l'apprentissage) peut faire qu'un classificateur ne fonctionnera pas dans des conditions optimales. Par exemple, un sur-ajustement du classificateur suite à l'utilisation d'un nombre excessif d'exemples ou encore l'utilisation de heuristiques d'apprentissage non validées ou sans fondement scientifique (*e.g.* , l'apprentissage répété sur un exemple tant que le classement n'est pas correct).

Des travaux de recherche continuent à être effectués à la recherche d'un filtrage parfait.

Yih Goodman et Hulten [270] ont étudié l'apprentissage d'un filtre dans des zones de faible taux de faux positifs, soit en attribuant des poids plus importants aux messages légitimes, soit en découpant le classement en deux étapes : dans chacune des étapes l'apprentissage serait réalisé sur des sous ensembles d'exemples différents. Malgré l'amélioration obtenue cette méthode semble être de mise en oeuvre difficile dans un contexte de filtrage en ligne.

Lynam et Cormack [181] ont expérimenté différentes combinaisons de 53 filtres indépendants et ont démontré que l'on peut obtenir des améliorations significatives avec des heuristiques simples de combinaison de juste un petit nombre de filtres. Wang et all [257] proposent un cadre d'étude et validation de méthodes de combinaison de classificateurs en vue d'obtention d'un ensemble optimal. La combinaison de classificateurs améliore la qualité de filtrage, au prix d'une complexification des filtres.

Ces sont deux exemples d'approches possibles, mais il reste des doutes sur la validité de ces approches : d'une part, ce sont des travaux validés toujours sur des données synthétiques, et d'autre part, la variabilité des contextes réels d'utilisation semble plus importante que le faible gain apporté par chacune de ces approches.

Martijn Grooten [119] a effectué un test comparatif de 21 produits commerciaux. Les plages d'efficacité obtenues sont résumées dans le tableau (12.6). Le corpus synthétique est constitué de messages légitimes en provenance de listes de diffusion et les spams en provenance de pièges à spam. Le corpus VirusBulletin est constitué de messages réels reçus par les employés de VirusBulletin. Ce tableau donne une idée (probablement optimiste) de l'efficacité des filtres anti-spam du commerce[3]. Ces filtres sont, pour la plupart, basés sur une combinaison de méthodes de classement et non pas une méthode unique.

[3]Nous avons vu dans le Chapitre 11 que les évaluations faites avec des corpus synthétiques donnent des résultats

Corpus	#Spams	Taux de vrai positifs	#Hams	Taux de faux positifs
Corpus synthétique	247315	98-99,8 %	2196	0-1,8 %
Corpus VirusBulletin	20829	87-97 %	1398	0-5,8 %

TAB. 12.6: Résumé de test comparatif de 21 produits commerciaux de filtrage de spam (*VirusBulletin*). Le corpus synthétique est constitué par des messages en provenance de listes de diffusion et pièges à spam, tandis que le corpus *VirusBulletin* est constitué de messages réels reçus par des employés de *VirusBulletin*

Le *filtrage imparfait de spam* est souvent mal perçu des utilisateurs de la messagerie électronique. D'une part ils constatent, tous les jours, l'apparition constante d'innovations dans les domaines de la communication électronique, inimaginables il y a quelques années, et d'autre part ils peuvent ne pas comprendre pourquoi ce problème n'est toujours pas résolu, alors qu'il est là depuis plus d'une décennie.

trop optimistes puisqu'ils ne sont pas représentatifs des flots réels de messages.

CHAPITRE 13

Comparaison de flots ou ensembles de messages

> Le savant doit ordonner ; on fait de la science avec des faits, comme on fait une maison avec des pierres ; mais une accumulation de faits n'est pas plus une science qu'un tas de pierres n'est une maison.
>
> *Henri Poincaré*, La Science et l'Hypothèse.

13.1 Introduction

Dans les chapitres précédents, une question est apparue entre les lignes et ignorée : la notion de *différence* on *ressemblance* entre boites aux lettres ou entre flots de messages.

Essayons de définir l'objectif (au moins un des) de cette réflexion. De façon caricaturale, le problème de classement consiste à, étant donné deux catégories d'objets et un objet susceptible d'appartenir à une d'entre elles, associer l'objet à la bonne catégorie.

Ce que nous souhaitons faire c'est être capables de décider si deux catégories données d'objets sont elles différentes ou pas et de quantifier que cette différence. Ou alors, ordonner N catégories d'objets selon leur similarité ou dissimilitude des catégories par rapport à une catégorie de référence.

Ceci est directement applicable dans le problème qui nous concerne, que nous posons de la façon suivante. Nous avons deux catégories de messages (A_1 et A_2), une troisième catégorie (B) et des classificateurs permettant de classer individuellement des messages des premières classes par rapport à la troisième. Une catégorie A_{1+2} est construite à partir d'un mélange de A_1 et A_2. Nous construisons un classificateur permettant de classer des messages en deux classes : A_{1+2} et B.

Notre problème est d'identifier les caractéristiques de A_1, A_2 et B qui feraient que le classificateur construit pour la classe issue du mélange aurait une efficacité encore comparable à celle des classes individuelles. Il s'agit d'un problème essentiel si nous souhaitons être capables de définir les frontières entre les communautés identifiées dans le Chapitre 7.

Intuitivement, une étape pour résoudre ce problème est la capacité de pouvoir comparer des catégories d'objets, de quantifier leur différence et l'augmentation de la diversité des ensembles combinés.

Une approche intuitive et naïve pour comparer deux ensembles de messages consiste à chercher les différences telles que perçues par les individus : les différences dans le contenu, le sujet, les mots, la mise en forme, les fichiers attachés, ... Bref, comparer les messages, mot par mot. Cette approche a peu d'intérêt puisque que les résultats ne sont ni quantifiables ni significatifs.

En fait, la *différence* pertinente n'est pas celle perçue par un être humain, mais celle perçue par le classificateur, qui n'accède qu'à une représentation des messages.

Dans ce chapitre nous présentons des réflexions sur la comparaison de flots ou ensembles de messages. En particulier, nous rappelons quelques idées issues du domaine de la Théorie d'Information qui justifieraient l'utilisation des divergences entre distributions de probabilité pour comparer des ensembles de messages soumis à classement par un algorithme du type génératif.

13.2 Les approches pour comparer des ensembles de messages

Comme il a été dit dans l'introduction, la comparaison pertinente n'est pas celle ressentie par un être humain, mais celle perçue par le classificateur. Il y a trois cas à considérer. Dans le premier cas, aucune information sur classificateur et la représentation interne des messages n'est pas accessible. Dans les deux autres, il convient de faire une distinction entre les classificateurs génératifs (*e.g.* le classificateur Bayésien Naïf et les classificateurs à compression de données) et les classificateur discriminants (*e.g.* , le classificateur à Régression Logistique, les SVM ou le Perceptron).

Dans ce chapitre nous traitons seulement le cas des classificateurs génératifs. Les sections suivantes présentent les idées de principe de la comparaison d'ensembles de messages soumis à un classificateur de ce type.

13.2.1 Les boîtes noires

Dans le chapitre 10, nous avons examiné trois flots de messages - des spams - nous sommes arrivés à la conclusion que ces flots ont une certaine ressemblance. En effet, dans les expérimentations de ce chapitre, rien n'a été supposé ni sur le classificateur, ni sur les flots de messages. Les classificateurs sont vus comme des boîtes noires. Les conclusions résultent uniquement de mesures indirectes telles la différence ressentie dans l'efficacité de classement, ou des caractéristiques déduites à l'aide d'outils particuliers (*e.g.* les variogrammes, les corrélogrammes ou périodogrammes) appliqués à des séries temporelles.

13.2.2 Les classificateurs génératifs

Dans les classificateurs de ce type (*e.g.* , le Bayésien Naïf), on associe un modèle à chaque classe. Le modèle correspond à la distribution de probabilités empirique des termes. L'opération de classement consiste à vérifier, pour l'objet à classer, quelle classe serait la plus susceptible de produire cet objet. Dans le cas du classificateur Bayésien Naïf, cela revient à évaluer une "distance" entre l'objet et chacune des distributions de probabilité. Cette distance peut être posée en termes d'un rapport de vraisemblance ou d'une divergence de Kullback-Leibler. Ainsi, on peut penser que la comparaison entre deux ensembles de messages, peut se réduire à évaluer la divergence de Kullback-Leibler entre les distributions de probabilité des termes.

13.2.3 Les classificateurs discriminants

Dans le cas de ces classificateurs, il n'y a pas de modèle associé à chaque classe. L'opération de classement est basé sur la recherche de la position relative de l'objet à classer par rapport à une surface de séparation entre les classes et d'attribuer l'une ou l'autre classe selon cette position relative. L'idée naïve consiste à vérifier si une telle surface existe. Une autre approche serait d'utiliser un ensemble de messages "témoin" que l'on sait différent des ensembles à comparer, et de vérifier si les surfaces de séparation entre les classes sont identiques ou proches autour des zones de concentration des objets. Cette comparaison peut ne pas etre triviale pour des classificateur non linéaires, mais dans le cas contraire, il suffirait de vérifier la distance entre des hyperplans de délimitation.

13.3 Le divergences entre distributions de probabilité

Dans cette section nous passons en revue quelques coefficients de divergence entre distributions de probabilité. Ce sont des mesures qui, du point de vue de la théorie d'information, ont une interprétation en rapport avec des bornes de taux d'erreur en teste d'hypothèses et applications de classement, comme nous le verrons dans la section suivante. Bien entendu, on considère que les ensembles de messages peuvent être représentés par des distributions empiriques de probabilité des termes.

Dans la bibliographie statistique, des nombreux coefficients ont été suggérés pour évaluer la ressemblance entre distributions de probabilité ou la facilité de les distinguer (manque de ressemblance), avec des dénominations différentes selon l'application. Nous utilisons la dénomination "divergence" et non pas métrique puisque, en général, ce ne sont pas des métriques dans le sens usuel. Kullback [156] a suggéré "*coefficient de divergence* d'une distribution par rapport à une autre". Une métrique est définie comme suit :

Définition 13.1. *Métrique ou Distance* [113] Soit $\mathcal{X}$ un ensemble quelconque. Une fonction $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}^+$, l'ensemble des nombres réels, est dite une ***métrique*** en $\mathcal{X}$ si :

1. $d(x, y) \geq 0, \forall x, y \in \mathcal{X}$ (Non-Négativité) ;
2. $d(x, y) = d(y, x), \forall x, y \in \mathcal{X}$ (Symétrie) ;
3. $d(x, y) = 0$, si et seulement si $x = y$ (Séparation) ;
4. $d(x, y) + d(y, z) \geq d(x, z)$, $\forall x, y, z \in \mathcal{X}$ (Inégalité Triangulaire)

Si d est une métrique, $(\mathcal{X}, d)$ est dit un *espace métrique*. Si d satisfait (2) et (4), mais pas forcément (3), on dit que d est un *écart*.

Ali et Silvey [4] suggèrent que une divergence entre distributions de probabilité doit satisfaire une liste non exhaustive de quatre propriétés :

1. Le coefficient $d(P_1, P_2)$ doit être défini pour toute paire P_1 et P_2 de l'espace de probabilités partagé.
2. Si l'on suppose que $y = t(x)$ est une transformation mesurable d'un espace de mesure $(\mathcal{X}, \mathcal{F})$ vers un autre espace $(\mathcal{Y}, \mathcal{G})$ alors, on doit avoir :

   $$d(P_1, P_2) \geq d(P_1 t^{-1}, P_2 t^{-1})$$

   où $P_i t^{-1}$ est la mesure en $\mathcal{Y}$ correspondante de la mesure P_i en $\mathcal{X}$. Cette propriété, qui n'est pas intuitive, résulte de l'inégalité de traitement des données [73] qui démontre que aucune transformation des données, surtout celles d'agrégation, ne peut améliorer la qualité de l'inférence que l'on peut faire à partir des données de départ. Cette propriété a une autre interprétation équivalente. Supposons que $\{x_n; n = 1, 2, \dots\}$ est un processus stochastique et que P_1 et P_2 sont deux possibles distributions pour ce processus. Il est plausible que plus longues sont les observations faites sur le processus, meilleure sera notre capacité d'identifier la vraie distribution.
3. $d(P_1, P_2)$ doit assumer une valeur minimale quand $P_1 = P_2$ et sa valeur maximale quand $P_1 \perp P_2$. Selon cette propriété, la divergence augmente lorsque les distributions *"s'éloignent"*.
4. Soit θ un paramètre réel et $\{P_\theta; \theta \in (a, b)\}$ une famille de distributions $P_\theta(x)$ avec des rapports de vraisemblance monotones en x. Si $a < \theta_1 < \theta_2 < \theta_3 < b$, alors :

   $$d(P_{\theta_1}, P_{\theta_2}) \leq d(P_{\theta_1}, P_{\theta_3})$$

   Cette propriété résulte de la précédente dans le sens où le but de la divergence est d'évaluer la discrimination de deux distributions et, donc, augmenter avec cette capacité, si l'on considère que le rapport de vraisemblance en est un indicateur.

13.3.1 Divergence de Kullback-Leibler

Définition 13.2. Soient P_1 et P_2 deux distributions de probabilité, discrètes, en $\mathcal{X}$. La Divergence de Kullback-Leibler [156] [73, P. 18] de P_2 par rapport à P_1 est définie comme :

$$D(P_1 \parallel P_2) = \sum_{x \in \mathcal{X}} p_1(x) \cdot \log \frac{p_1(x)}{p_2(x)} \tag{13.1}$$

On considère les cas particuliers : $0 \cdot log\frac{0}{0} = 0$, $0 \cdot log\frac{0}{q} = 0$ et $p \cdot log\frac{p}{0} = \infty$.

La divergence de Kullback-Leibler (KL) est toujours non-négative et n'est nulle que quand les distributions P_1 et P_2 sont identiques partout dans $\mathcal{X}$.

$$D(P_1 \parallel P_2) = 0 \iff P_1(x) = P_2(x),\ \forall x \in \mathcal{X} \tag{13.2}$$

La divergence de Kullback-Leibler n'est pas symétrique et n'a pas la propriété de l'inégalité triangulaire. Certains auteurs (y compris Kullback [156]) ont étudié la J-divergence proposée initialement par Jeffreys [135], comme solution de remplacement symétrique de la divergence de Kullback-Leibler.

$$J(P_1 \parallel P_2) = \frac{1}{2}(D(P_1 \parallel P_2) + D(P_2 \parallel P_1)) \tag{13.3}$$

Néanmoins, la divergence de Kullback-Leibler est intéressante justement à cause de son asymétrie, puisque les problèmes de test d'hypothèses et de classement sont des problèmes intrinsèquement asymétriques, et cette propriété est préservée par la divergence de Kullback-Leibler.

Par contre, l'inconvénient majeur est qu'elle n'est pas bornée lorsqu'une des distributions assume des valeurs nulles dans une partie de son espace de probabilités (cas $p \log \frac{p}{0}$).

En théorie d'information, cette divergence est connue sous la dénomination de *Entropie Relative* et interprétée comme le nombre de bits additionnels nécessaires pour coder une information, utilisant une distribution de probabilité autre que celle qui a effectivement généré l'information. C'est le principe de fonctionnement des classificateurs à compression.

Nous verrons dans la section (13.4) des propriétés de la divergence KL qui justifient son utilisation en test de hypothèses. Lafferty utilise la divergence KL comme fonction risque à minimiser en recherche documentaire [160]. Le AIC (Akaike Information Criteria) est un critère permettant d'évaluer la qualité d'un modèle fondé sur l'utilisation de la divergence KL pour comparer des distributions de probabilité [154].

13.3.2 Divergence de Jensen-Shannon

La divergence de Jensen-Shannon cherche à symétriser la divergence de Kullback-Leibler et à palier l'inconvénient d'absence d'une borne supérieure. Cette divergence a été introduite implicitement par Wong [264] puis formalisée par Lin [175] en 1991. Elle est définie comme suit :

Définition 13.3. *Divergence de Jensen-Shannon* - Soient P_1 et P_2 deux distributions de probabilité, discrètes, en $\mathcal{X}$. La divergence de Jensen-Shannon est définie par :

$$\begin{aligned} JSD(P_1, P_2) &= D(P_1 \parallel \overline{P}) + D(P_2 \parallel \overline{P}) \\ \overline{P}(x) &= \frac{P_1(x) + P_2(x)}{2},\ x \in \mathcal{X} \end{aligned} \tag{13.4}$$

Cette divergence est donc, la somme des divergences de Kullback-Leibler entre les distributions en question et leur moyenne. Vu que $\overline{P}(x)$ n'est jamais nulle dans les points où au moins une des deux distributions n'est pas nulle, la divergence de Jensen-Shannon est toujours bornée et en fait elle est toujours inférieure à 1 [175].

Définition 13.4. *Divergence de Jensen-Shannon généralisée* - Soient $P_1, P_2, \ldots, P_N$ des distributions de probabilité en $\mathcal{X}$, et $\pi_1, \pi_2, \ldots, \pi_N$ des réels de somme unitaire :

$$\begin{aligned} JSD_\pi(P) &= \sum_{i=1}^{N} D(P_i \parallel \overline{P}) \\ \overline{P}(x) &= \sum_{i=1}^{N} \pi_i P_i(x),\ x \in \mathcal{X},\ \sum_{i=1}^{N} \pi_i = 1 \end{aligned} \tag{13.5}$$

Une interprétation de la divergence de Jensen-Shannon [107] résulte d'un modèle de commutation (switching model) où une information est générée par une parmi N sources. A chaque nouvelle information, la source émettrice est choisie aléatoirement avec probabilité π_i. La divergence de Jensen-Shannon correspond à la redondance minimale nécessaire pour encoder l'information résultante de la différence entre l'entropie moyenne des sources plus l'incertitude liée au choix de la source émettrice.

Un autre interprétation [107] concerne les modèles de mélange, que nous aborderons dans le chapitre (14). Dans ce contexte, la divergence de Jensen-Shannon est interprétée comme l'augmentation de l'entropie de la distribution de probabilité résultante du mélange, par rapport à l'entropie moyenne pondérée des composantes.

La racine carrée de la divergence de Jensen-Shannon est une métrique. Voir démonstration en [196].

13.3.3 Les *f-divergences*

La notion de *f-divergences* a été introduite par Csiszar [78] et Ali et Silvey [4] en même temps, mais développées indépendamment, comme une proposition d'unification des différentes mesures de divergence entre distributions de probabilité.

Définition 13.5. *f-divergence* - Soit $f(t)$ une fonction convexe définie pour $t > 0$, avec $f(1) = 0$ et $g(t)$ une fonction croissante. La *f-divergence* d'une distribution P par rapport à Q est définie par :

$$\begin{aligned} D_f((P \parallel Q) &= g\left(E_Q f\left(\frac{P(X)}{Q(X)}\right)\right) \\ &= g\left(\sum_{x \in \mathcal{X}} Q(x) f\left(\frac{P(x)}{Q(x)}\right)\right) \end{aligned} \tag{13.6}$$

On considère, $0\ f\left(\frac{0}{0}\right) = 0$, $f(0) = \lim_{t \to 0} f(t)$, $0\ f\left(\frac{a}{0}\right) = \lim_{t \to 0} t f(\frac{a}{t}) = a \lim_{u \to \infty} \frac{f(u)}{u}$.

Remarque 13.1. Certaines références incluent la fonction croissante $g(t)$ dans la définition des *f-divergences* (*e.g.* [14]) alors que les références originales l'omettent (*e.g.* [4] [79]) (dont la forme implicite triviale $g(t) = t$). Cette variante permet d'intégrer d'autres mesures dans la famille des f-divergences, telles la divergence de Chernoff et de Battacharyya. Nous l'avons maintenu dans le seul but de garder en mémoire un possible lien avec ces autres mesures. Néanmoins, il ne faut pas perdre de vue que les démonstrations des propriétés que l'on trouve dans la bibliographie [79] considèrent une fonction $g(t)$ triviale et ne sont pas forcément valables lorsque la fonction $g(t)$ n'est pas triviale.

La table (13.1) présente des exemples de divergences que l'on peut traiter comme une *f-divergence*.

Les *f-divergences* ont été intensivement étudiées [79] [196] [197] [201] [252] [174] avec démonstration des propriétés communes.

Seules quelques unes des *f-divergences* sont des vraie métriques, comme par exemple, la Divergence de Hellinger, lorsque $f(t) = (1 - \sqrt{t})^2$ [88].

Mesure	$f(t)$	$g(t)$	$D_f(P \parallel Q)$
Divergence de Kullback-Leibler - $D(P \parallel Q)$	$t \log t$	t	$\sum_{x\in\mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}$
Divergence de Kullback-Leibler - $D(Q \parallel P)$	$-\log t$	t	$\sum_{x\in\mathcal{X}} Q(x) \log \frac{Q(x)}{P(x)}$
Divergence de Jeffreys	$(t-1)\log t$	t	$\sum_{x\in\mathcal{X}} (P(x)-Q(x)) \log \frac{P(x)}{Q(x)}$
Divergence de $\mathcal{X}^2$ (Pearson)	$(t-1)^2$	t	$\sum_{x\in\mathcal{X}} \frac{(P(x)-Q(x))^2}{Q(x)}$
Divergence de Chernoff	$-x^{1-r}$, $0 \leq r \leq 1$	$-\log(-t)$	$-\log\left(\sum_{x\in\mathcal{X}} P^r(x)Q^{1-r}(x)\right)$
Divergence de Hellinger (1)	$(\alpha-1)^{-1}(t^\alpha-1)$	t	$1-\sum_{x\in\mathcal{X}} \sqrt{P(x)Q(x)}$
Divergence de Hellinger (2)	$1-\sqrt{t}$, $\left(\alpha=\frac{1}{2}\right)$	t	$1-\sum_{x\in\mathcal{X}} \sqrt{P(x)Q(x)}$
Divergence de Battacharyya	$-\sqrt{t}$	$-\log(-t)$	$-log\left(\sum_{x\in\mathcal{X}} \sqrt{P(x)Q(x)}\right)$
Distance variationnelle (Matsushita)	$\lvert t-1\rvert$	t	$\sum_{x\in\mathcal{X}} \lvert P(x)-Q(x)\rvert$
Information généralisée	$\lambda^{-1}(x^\lambda-1)$	t	$\frac{1}{\lambda}\sum_{x\in\mathcal{X}} \left\{\left(\frac{P(x)}{Q(x)}\right)^\lambda - 1\right\}$

TAB. 13.1: Exemples de représentation de mesures de dissimilarité entre distributions de probabilité, sous la forme de *f-divergences* [4] [79] [154] [14]

13.3.4 Le cosinus et la distance euclidienne

Le *cosinus* et la *distance euclidienne* sont, avec la divergence de Kullback-Leibler, des mesures de similitude utilisées en recherche documentaire pour comparer des documents ou des ensembles de documents (clustering) [182, p. 344]. Ces mesures apparaissent naturellement dans le modèle VSM[1]. Les documents, ou classes de documents, sont représentés par des vecteurs où les composantes sont les fréquences de chaque terme.

La similarité par le cosinus est définie comme :

$$sim(\overline{P_1}, \overline{P_2}) = \cos(\overline{P_1}, \overline{P_2}) = \frac{<\overline{P_1}, \overline{P_2}>}{|\overline{P_1}||\overline{P_2}|} \tag{13.7}$$

L'intérêt du *cosinus* la normalisation naturelle de la longueur des documents. Mais les inconvénients sont le faible pouvoir discriminant pour des documents assez proches et l'absence de rapport entre la mesure et des taux d'erreur.

La distance euclidienne est définie comme :

$$d(P_1, P_2) = \sqrt{\sum_{x\in\mathcal{X}} (P_1(x) - P_2(x))^2} \tag{13.8}$$

Dans le cas où les vecteurs définissant les distributions de probabilité sont normalisés, l'ordre (ranking) généré par un ensemble de distributions de probabilité utilisant la distance euclidienne ou le cosinus sont identiques [182, p. 120].

La distance euclidienne est une métrique.

13.4 Propriétés des Divergences

Dans cette section nous présentons quelques propriétés suggérant le lien entre les divergences entre distributions de probabilité et la problématique de test d'hypothèses, en particulier la

[1] VSM - Vector Space Model

probabilité asymptotique d'erreur.

13.4.1 Le lemme de Neymann-Pearson

Parmi les différentes façons de présenter le Lemme de Neymann-Pearson [168] [260] [73], celle que l'on trouve dans les textes de théorie d'information et des grandes déviations [73], fait apparaitre naturellement les liens entre la comparaison de distributions de probabilité et la problématique de classement qui nous intéresse.

Théorème 13.4.1 (Lemme de Neymann-Pearson)**.** *Soit $X_1, X_2, ..., X_n$ une suite de variables aléatoires tirées i.i.d. selon une distribution Q. Considérons le problème de test d'hypothèses qui consiste à décider si $Q = P_1$ ou si $Q = P_2$. Q, P_1 et P_2 définies dans le même espace de probabilités. Pour $T \geq 0$ définissons la région $A_n(T)$ de décision :*

$$A_n(T) = \left\{ x^n \in \mathcal{X}^n : \frac{P_1(x_1, x_2, \ldots, x_n)}{P_2(x_1, x_2, \ldots, x_n)} > T \right\} \tag{13.9}$$

où $\frac{P_1(a)}{P_2(a)}$ est le rapport de vraisemblance de a sous P_1 et P_2.

Soit α^* la probabilité d'erreur associée au choix de P_2 alors que le bon choix est P_1 et β^* le cas contraire :

$$\alpha^* = P_1^n(A_n^c(T)), \quad \beta^* = P_2^n(A_n(T)) \tag{13.10}$$

Soit B_n une autre région quelconque, avec des probabilités d'erreur α et β, alors :

$$\alpha \leq \alpha^* \Longrightarrow \beta \geq \beta^* \tag{13.11}$$

Démonstration. Voir [73, p. 376] □

Le lemme de Neyman-Pearson permet de définir un test optimal dans le sens suivant. Si l'on fixe l'une des deux probabilités d'erreur (e.g. α^*) et si on considère toutes les autres possibles régions de décision, aucune d'entre elles ne minimisera, en même temps α et β. Cette région définit la région de $\mathcal{X}^n$ où la décision optimale est la distribution P_1.

Le test du lemme de Neyman-Pearson sur deux hypothèses est la comparaison du rapport de vraisemblance avec un seuil, de la forme :

$$\frac{P_1(x_1, x_2, \ldots, x_n)}{P_2(x_1, x_2, \ldots, x_n)} \gtrless T \tag{13.12}$$

Dans le contexte d'une application de classement, le critère (13.12) est interprété comme une mesure de distance entre distributions de probabilité [156, p. 86] [73, p. 378]. Soit $L(\cdot)$ le log du rapport de vraisemblance[2] :

$$\begin{aligned} L(x_1, x_2, \ldots, x_n) &= \log \frac{P_1(x_1, x_2, \ldots, x_n)}{P_2(x_1, x_2, \ldots, x_n)} \\ &= \sum_{i=1}^{n} \log \frac{P_1(x_i)}{P_2(x_i)} \end{aligned} \tag{13.13}$$

[2]Ce passage utilise l'hypothèse d'indépendance statistique des termes.

Soit P_{X^n} la distribution de probabilité empirique de la séquence $x_1, x_2, \ldots, x_n$

$$\begin{aligned} L(x_1, x_2, \ldots, x_n) &= \sum_{a \in \mathcal{X}} n \; P_{X^n}(a) \; \log \frac{P_1(a)}{P_2(a)} \\ &= \sum_{a \in \mathcal{X}} n \; P_{X^n}(a) \; \log \frac{P_1(a) P_{X^n}(a)}{P_2(a) P_{X^n}(a)} \\ &= \sum_{a \in \mathcal{X}} n \; P_{X^n}(a) \; \log \frac{P_{X^n}(a)}{P_2(a)} \\ &\quad - \sum_{a \in \mathcal{X}} n \; P_{X^n}(a) \; \log \frac{P_{X^n}(a)}{P_1(a)} \\ &= nD(P_{X^n} \parallel P_2) - nD(P_{X^n} \parallel P_1) \end{aligned} \tag{13.14}$$

Donc, le test 13.12 est équivalent à :

$$D(P_{X^n} \parallel P_2) - D(P_{X^n} \parallel P_1) \lessgtr \frac{1}{n} \log T \tag{13.15}$$

c.à.d. , la décision optimale consiste à choisir, parmi les deux distributions de probabilité candidates, celle dont la "distance" avec la distribution de probabilité empirique de l'échantillon est la plus faible.

13.4.2 Lemme de Chernoff-Stein

Théorème 13.4.2 (AEP - Propriété d'Equipartition Asymptotique[3] de l'entropie relative [73])**.** *Soit $X_1, X_2, \ldots, X_n$ une séquence de variables aléatoires tirées i.i.d. selon une loi $P_1(x)$ sur $\mathcal{X}$ et $P_2(x)$ n'importe quelle autre distribution en $\mathcal{X}$. Alors, le rapport de vraisemblance converge en probabilité vers la divergence de Kullback-Keibler entre les distributions des deux classes.*

$$\frac{1}{n} \log \frac{P_1(X_1, X_2, \ldots, X_n)}{P_2(X_1, X_2, \ldots, X_n)} \quad \xrightarrow{P} \quad D(P_1 \parallel P_2) \tag{13.16}$$

Démonstration. Il s'agit d'un résultat immédiat de la loi faible des grands nombres [73, p. 380] :

$$\begin{aligned} \frac{1}{n} \log \frac{P_1(a) P_{X^n}(a)}{P_2(a) P_{X^n}(a)} &= \frac{1}{n} \log \frac{\prod_{i=1}^n P_1(X_i)}{\prod_{i=1}^n P_2(X_i)} \\ &= \frac{1}{n} \sum_{i=1}^n \log \frac{P_1(X_i)}{P_2(X_i)} \\ &\xrightarrow{P} E_{P_1} \log \frac{P_1(X_i)}{P_2(X_i)} \\ &= D(P_1 || P_2) \end{aligned} \tag{13.17}$$

□

Théorème 13.4.3 (Lemme de Chernoff-Stein)**.** *Soit $X_1, X_2, \ldots, X_n$ une séquence de variables aléatoires tirées i.i.d. $\sim Q$. Considère le test d'hypothèses entre deux alternatives, $H_0 : Q = P_1$ et $H_1 : Q = P_2$, avec $D(P_1 \parallel P_2) < \infty$. Soit $A_n \subseteq \mathcal{X}^n$ une région d'acceptation pour l'hypothèse $Q = P_1$. Les probabilités d'erreur sont :*

$$\alpha^* = P_1^n(A_n^c(T)), \quad \beta^* = P_2^n(A_n(T)) \tag{13.18}$$

Pour $0 < \epsilon < \frac{1}{2}$, définir :

[3] AEP - Asymptotic Equipartition Property

$$\beta_n^\epsilon = \min_{\substack{A_n \subseteq \mathcal{X}^n \\ \alpha_n < \epsilon}} \beta_n \tag{13.19}$$

Alors,

$$\begin{aligned} \lim_{n\to\infty} \frac{1}{n} \log \beta_n^\epsilon &= -D(P_1 \parallel P_2) \\ \lim_{n\to\infty} \beta_n^\epsilon &= 2^{-nD(P_1 \| P_2)} \end{aligned} \tag{13.20}$$

Démonstration. Voir [73, p. 384]. □

Ces deux résultats sont très intéressants puisqu'ils démontrent l'existence d'un rapport liant la divergence de Kullback-Leibler entre les distributions de probabilité des termes dans les classes, la région de décision optimale et des bornes des taux d'erreur de classement. À noter qu'il s'agit de rapports asymptotiques, pour des séquences dont la longueur tend vers infini.

Ces deux propriétés justifient l'utilisation d'une des divergences (Kullback-Leibler) pour évaluer la dissimilarité d'ensembles d'objets. Les propriétés des divergences et leurs applications dans les applications de classement et test d'hypothèses ont été largement étudiées. Voir, par exemple, Devroye [88] Beirlant [16] Pardo [201] Vajda et Osterreicher [196] et Csiszár [79].

13.5 Résultats Préliminaires

Les expérimentations préliminaires avec les données disponibles que nous avons semblent montrer, au moins qualitativement, que la comparaison des distributions de probabilité des termes reflète effectivement la différence ressentie entre ensembles de messages.

Les distributions empiriques de probabilité ont été construites à partir de la représentation des messages utilisée par le classificateur bayésien *naif* du logiciel *j-chkmail*. Les termes sont les mots et bi-mots.

Nous avons utilisé quatre des mesures présentées dans ce chapitre pour comparer des ensembles de messages de trois informaticiens : *joe*, *jmv* et *mg* - voir résultats dans la Table 13.2. Les deux points intéressants de ces résultats sont : les quatre mesures ont placé les ensembles dans le même ordre et les ensembles les plus proches, *joe* et *mg*, correspondent, effectivement, aux personnes dont les profils professionnels sont aussi les plus proches.

La Table 13.3 présente les résultats de comparaison des flots de messages de l'auteur, regroupés par mois, de mars 2008 à juin 2008 : *hams* (H-08mm) et *spams* (S-08mm), utilisant la divergence de Kullback-Leibler. On remarque que les différences intra-classe sont inférieurs aux différences inter-classe.

La Figure 13.1 (13.1a à 13.1c) montre l'évolution temporelle des flots de messages utilisés dans les expérimentations du Chapitre 10, selon les divergences de Kullback-Leibler, Jensen-Shannon et distance euclidienne, évaluées par rapport à l'ensemble de messages de départ.

13.6 Discussion et Conclusions

Dans ce chapitre nous avons posé le problème de la quantification de la ressemblance (ou dissemblance) entre ensembles de messages. Nous avons identifié trois approches pour traiter ce problème, la première considérant le système de filtrage comme une boite noire et les deux autres, dépendantes du type de classificateur, nécessitant une connaissance de l'algorithme de classement et de la représentation des messages.

Nous avons présenté quelques résultats issus de la théorie de l'information justifiant l'utilisation des divergences entre distributions de probabilité pour évaluer la différence entre ensembles de messages dans le cas des algorithmes génératifs. Les résultats expérimentaux obtenus semblent cohérents avec l'attendu.

	h-jmv	h-joe	h-mg
h-jmv	0.0000	2.1459	2.1323
h-joe	1.8279	0.0000	1.3877
h-mg	1.5156	1.3509	0.0000

(a) Divergence de Kullback-Leibler

	h-jmv	h-joe	h-mg
h-jmv	0.0000	0.6726	0.5929
h-joe	0.6726	0.0000	0.4660
h-mg	0.5929	0.4660	0.0000

(b) Divergence de Jensen-Shannon

	h-jmv	h-joe	h-mg
h-jmv	0.0000	1.3974	1.2009
h-joe	1.3974	0.0000	0.8809
h-mg	1.2009	0.8809	0.0000

(c) Divergence de Battacharyya

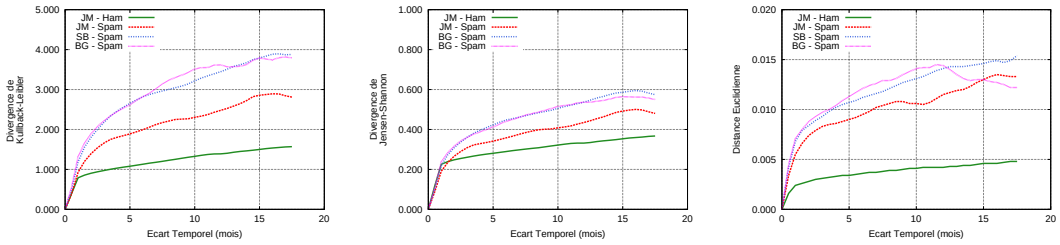
	h-jmv	h-joe	h-mg
h-jmv	0.0000	0.0063	0.0064
h-joe	0.0063	0.0000	0.0050
h-mg	0.0064	0.0050	0.0000

(d) Distance Euclidienne

TAB. 13.2: Dissimilarités entre les boîtes aux lettres de trois informaticiens évaluées à l'aide de divergences et distance euclidienne

	H-0803	H-0804	H-0805	H-0806	S-0803	S-0804	S-0805	S-0806
H-0803	0.0000	0.7357	0.7817	0.8103	2.8219	3.0093	3.0501	3.0562
H-0804	0.7540	0.0000	0.7166	0.7805	2.8360	3.0163	3.0567	3.0614
H-0805	0.8289	0.7365	0.0000	0.7273	2.8296	3.0127	3.0491	3.0578
H-0806	0.8615	0.8018	0.7364	0.0000	2.8597	3.0499	3.0855	3.0827
S-0803	3.1116	3.1209	3.1348	3.1521	0.0000	0.7899	1.2674	1.4401
S-0804	3.3431	3.3445	3.3630	3.3788	1.0430	0.0000	0.8971	1.3293
S-0805	3.2936	3.3020	3.3131	3.3240	1.4439	0.8802	0.0000	0.7894
S-0806	3.2122	3.2137	3.2322	3.2343	1.5344	1.2690	0.8378	0.0000

TAB. 13.3: Évolution temporelle des *hams* et *spams* de l'auteur, entre mars et juin 2008, évaluée par la divergence de Kullback-Leibler.



(a) Div Kullback-Leibler (b) Div Jensen-Shannon (c) Distance Euclidienne

FIG. 13.1: Évolution temporelle des flots de messages utilisés dans le Chapitre 10 par les mesures de divergence de Kullback-Leibler, Jensen-Shannon et distance euclidienne.

Néanmoins, si ces résultats semblent cohérents, la démarche de l'expérimentation n'a pas tenu compte d'un nombre de détails du fonctionnement de l'algorithme de classement. Par exemple :

- **modèle de génération des messages** : dans les développements issus de la théorie d'information, le modèle utilisé est un tirage aléatoire avec remise, tandis que dans la plupart des algorithmes de classement, il s'agit d'un tirage sans remise, puisque seule la présence d'un terme compte, et non pas le nombre d'occurrences ;
- **sélection d'attributs** : les algorithmes de classement usuels sélectionnent les attributs les plus discriminants trouvés dans l'objet à classer ;
- **apprentissage supervisé versus actif** : les résultats issus de la théorie d'information se basent sur tous les exemples disponibles tandis que dans l'apprentissage actif, seuls les exemples informatifs sont pris en compte. Cela fait que l'aspect probabiliste de la distribution des objets n'est pas respecté. Le contour de séparation des classes n'a plus un caractère probabiliste e devient plutôt géométrique.

Ces trois exemples ne sont pas, bien entendu, exhaustifs et la validité des développements reste à démontrer. Néanmoins, une réflexion sur les méthodes permettant de comparer, du point de vue du classificateur, des flots de messages reste intéressante et mérite être poursuivie.

13.7 Notes Historiques et Bibliographiques

La divergence de Kullback-Leibler a été proposée par Kullback et Leibler en 1951 [157] [156], sous la dénomination *information discriminante* (information for discrimination). Dans cet article, les auteurs présentent, dans le cadre même cadre, leur mesure ainsi que celle introduite par H. Jeffreys quelques années auparavant [135] et qui était une version symétrique de leur mesure d'*information discriminante* obtenue par la moyenne des mesures dans les deux sens. Kullback [156] suggère la minimisation de cette mesure comme un principe en inférence statistique.

Plusieurs mesures de dissimilarité avaient été proposées avant celle de Kullback et Leibler : les divergences de Chernoff, Hellinger, Mahalanobis, Battacharyya, pour ne citer que quelques unes. La nouveauté apportée par Kullback et Leibler est le lien, intentionnel, entre cette nouvelle mesure et la Théorie d'Information, introduite peu de temps avant par la célèbre publication de Claude E. Shannon [236], qui proposait l'entropie comme mesure de quantité d'information. Dans le préface de son livre [156], Kullback écrit :

> Information theory is a branch of the mathematical theory of probability and mathematical statistics. As such, it can be and is applied in a wide variety of fields. Information theory is relevant to statistical inference and should be of basic interest to statisticians. Information theory provides a unification of known results, and leads to natural generalizations and the derivation of results.

Dans cette même publication, Kullback démontre l'existence d'une relation entre la *information discriminante* et l'information de Fischer.

En 1966 et 1967, Imre Csiszár [78] et Ali et Silvey [4] ont introduit simultanément et indépendamment les *f-divergences*, comme un moyen d'unifier les différentes mesures de dissimilarité, comme une application de la théorie des grandes déviations.

La plupart de ces mesures n'étant pas des métriques, plusieurs propositions de mesures dérivées ont été faites dans le but de trouver des mesures de dissimilarité qui étaient des vraies métriques dans le sens topologique. Néanmoins, cet asymétrie ne fait que reproduire, légitimement, l'asymétrie des problèmes test d'hypothèses et classement.

On trouvera une présentation plus complète (et didactique) des propriétés de ces mesures, et de leur application dans les problèmes de reconnaissance de formes, dans [88].

Parmi les mesures symétriques intéressantes, on trouve la Divergence de Jensen-Shannon, développée par Lin [175] et qui a fait objet de plusieurs publications. Osterreicher et Vajda ont démontré que la racine carrée de cette divergence est une vraie métrique [196], [197]. Une autre preuve a été fournie par Endres et Schindelin [98]. Topsoe et Fuglede [252] [107] ont démontré plusieurs propriétés de cette mesure, en particulier des rapports (inégalités) avec autres divergences telles la divergence de Hellinger et la divergence variationnelle.

CHAPITRE 14

Modèles de Mélange Fini

> You give 100 percent in the first half of the game, and if that isn't enough in the second half you give what's left.
>
> *Yogi Berra*

14.1 Introduction

Les modèles de mélange et commuté apparaissent naturellement à différents niveaux dans la constitution des boîtes aux lettres et les flots de messages.

La plupart des utilisateurs de la messagerie électronique a l'habitude de classer ses messages dans des dossiers selon, par exemple, le sujet ou l'expéditeur. Quelque soit le critère de classement, chaque dossier a ses particularités qui font que les messages à l'intérieur de chaque dossier ont des caractéristiques communes. La boîte aux lettres d'un utilisateur de la messagerie est l'*union* de plusieurs dossiers.

Si on remonte à un niveau plus haut, comme par exemple, un département d'une université, on peut considérer que la *boîte aux lettres collective du département* est l'union des boîtes aux lettres des différents utilisateurs. De cette façon, on pourrait représenter cette *boîte aux lettres collective* comme une composition pondérée, une combinaison linéaire, des différents sujets traités par les différents utilisateurs.

A un niveau encore plus haut, on peut voir l'Internet entier comme un réseau avec des noeuds de routage où des flots de messages sont combinés (mélangés) ou éclatés.

Ces modèles sont directement utilisables dans les applications utilisant des algorithmes génératifs de classement tels le Bayésien Naïf. L'idée est qu'une boîte aux lettres peux être représentée par la distribution de probabilité de ses unités textuelles et que l'on peut combiner, successivement, les distributions que l'on trouve à chaque niveau pour retrouver celles des niveaux plus élevés.

À l'exception du problème trivial de déterminer la distribution résultante sachant les composantes et les poids de chacune, les problèmes à résoudre avec de modèles de mélange et de commutation ont souvent un coût de traitement élevé. Néanmoins, l'abordage de la problématique du filtrage collectif de spam en tant que modèle de mélange ou de commutation a le mérite de fournir quelques réponses et conclusions intéressantes.

14.2 Modèles de mélange et commuté

Définition 14.1. [106] Considère une variable (ou vecteur) aléatoire X en un espace de probabilités $\mathcal{X}$, discret ou continu. X est dite résulter d'un mélange fini de N distributions si la fonction de distribution $P(X)$ prend la forme d'un mélange, pour tout $X \in \mathcal{X}$:

$$P(x) = \sum_{i=1}^{N} \eta_i P_i(x),\ \eta_i \geq 0,\ \sum_{i=1}^{N} \eta_i = 1 \tag{14.1}$$

On considère souvent que les distributions composantes appartiennent à une même famille de distributions paramétriques indexées par un paramètre Θ, mais elles peuvent aussi provenir de distributions non paramétriques partageant le même espace de probabilités.

Pour décrire le modèle commuté, reprenons le modèle de mélange fini décrit par l'équation (14.1).

En fait, ces deux modèles ont des propriétés assez similaires et sont, assez souvent, abordés ensemble [106] [187] [23]. La différence réside dans la définition du processus de génération des objets d'étude. Dans le modèle de mélange les objets sont générés selon la distribution résultante du mélange, tandis que dans le modèle commuté, il y a, d'abord, un choix au hasard d'une distribution (les coefficients de mélange correspondent à une distribution de probabilités) et ensuite la génération de l'objet, selon la distribution choisie par le premier tirage.

Le modèle commuté semble, à priori, plus adapté à la modélisation d'un flot de messages puisqu'il s'agirait, d'abord, de choisir une catégorie (ham, spam), une sous-catégorie (appel présentation, message de service, pornographie, médicaments, ...) pour ensuite générer le message selon la sous-catégorie choisie. Néanmoins, ce mode de fonctionnement nécessite, lors de l'opération de classement, de pouvoir identifier la composante du mélange, ce qui n'est pas forcément une tâche aisée. Une contrainte nécessaire pour qu'un mélange soit identifiable est que des paramètres de mélange différents doivent correspondre à des distributions différentes [106, p. 14].

Aussi, lorsque la dimension du problème est élevée, les méthodes fondées sur l'estimation de la vraisemblance deviennent rapidement intraitables [106, p. 43] [212, p. 11], puisque exprimée comme un produit de sommes. La fonction de vraisemblance d'une séquence de dimension N, $x_1, x_2, \ldots, x_N$, générée par un mélange de K composantes a la forme [106, p. 43] :

$$L(x_1, x_2, \ldots, x_N) = \prod_{i=1}^{N} \sum_{k=1}^{K} \eta_k P_k(x_i) \tag{14.2}$$

14.3 Discussion et perspectives

Il ne semble pas évident que l'on puisse se baser sur les modèles de mélange pour effectuer du classement de messages électroniques, à cause de la lourdeur des traitements. Néanmoins, ces modèles apparaissent naturellement dans la constitution des flots et on peut penser qu'ils puissent être utiles plutôt dans un but d'approfondissement de la connaissance. On peut citer deux exemples d'applications.

Dans le Chapitre 10 nous avons émis l'hypothèse que le flot de *spams* arrivant dans une boite aux lettres pourrait résulter de la combinaison d'un petit nombre de flots indépendants. On peut imaginer qu'un modèle de mélange pourrait être utilisé pour identifier et évaluer les coefficients de mélange en jeu.

Une deuxième application possible pourrait être l'étude de la diversité du flot composé, par exemple, dans une université. Nous avons démontré empiriquement dans le Chapitre 11 que l'efficacité d'un classificateur mutualisé diminue au fur et à mesure que l'on ajoute des composantes différentes, donc la diversité de la population. Il serait alors intéressant de comprendre comment et de pouvoir établir des limites d'utilisabilité. L'entropie d'une distribution est une mesure "grossière", de la diversité d'une distribution de probabilité. On peut démontrer qu'il est possible d'évaluer l'entropie d'une distribution issue d'un mélange à partir des composantes du

mélange. De même, on peut aussi évaluer certaines divergences (*e.g.* , Kullback-Leibler) entre distributions issus d'un mélange, aussi à partir de ces composantes[1].

[1]La démonstration a été omise, puisque longue et l'intérêt de la démarche reste à démontrer.

Sixième partie

Conclusions

CHAPITRE 15

Conclusions

> The important thing is not to stop questioning. Curiosity has its own reason for existing. One cannot help but be in awe when he contemplates the mysteries of eternity, of life, of the marvelous structure of reality. It is enough if one tries merely to comprehend a little of this mystery every day. Never lose a holy curiosity.
>
> *Albert Einstein*

15.1 Résultats

Notre premier objectif, au début de cette thèse, était l'étude du classement mutualisé de messages électroniques dans une communauté telle une université ou plus généralement, un établissement d'enseignement et recherche avec une population, certes, hétérogène mais avec des points communs.

L'expérience préalable et les premiers travaux ont montré que quasiment tous les algorithmes de classement et méthodes d'apprentissage usuels avaient déjà été expérimentés pour le filtrage de *spam*. Néanmoins, la connaissance de la *matière manipulée* - les caractéristiques des messages - restait, à notre avis, insuffisante.

Pour mener à bien notre objectif initial, il nous a semblé plus intéressant de, avant toute autre chose, augmenter cette connaissance que de rechercher des améliorations aux algorithmes et méthodes déjà existantes.

Le problème de classement de messages électroniques diffère de celui de classement de documents textuels génériques dans le sens où les messages électroniques contiennent la partie utile mais aussi des méta-informations (e.g. des informations de mise en forme, des informations de routage ou des fichiers attachés) et une diversité linguistique bien plus large que celles des documents textuels génériques. Par ailleurs, l'observation des résultats dans les configurations diverses a montré que les méta-informations génèrent des attributs plus pertinents pour le classement que le contenu effectif des messages.

Après avoir étudié des travaux déjà publiés concernant le filtrage de spams, nous avons défini une architecture de filtrage susceptible d'être utilisée dans le contexte initial de cette thèse, avec la contrainte d'être encore simple mais performant. Nous avons utilisé un algorithme discriminant linéaire assez simple avec apprentissage en ligne (approximation stochastique) et apprentissage actif (marge statique).

Nous avons utilisé cette architecture, tout en désactivant l'apprentissage, pour étudier l'évolution temporelle de trois flots de *spams* et un flot de *hams*. Cette expérimentation nous a permis

de mettre en évidence des caractéristiques intéressantes des flots et de démontrer que les deux classes de messages n'ont pas les même caractéristiques.

Dans un contexte simulé de filtrage utilisant des ensembles de messages réels et synthétiques, tels ceux souvent utilisés dans les travaux publiés, nous avons pu démontrer que ces derniers ne correspondent pas à des situations réelles et donnent des résultats excessivement optimistes, en particulier par l'absence de *messages gris (gris mails)*, les messages pouvant être appréciés différemment selon le destinataire (*e.g.* les messages publicitaires).

La simulation de classement de messages réels, utilisant notre proposition d'architecture de filtrage, a donné des résultats que nous considérons très bons (taux de faux positifs de l'ordre de 0,3 %, taux de faux négatifs de l'ordre de 0,03 % et 1-AUC de l'ordre de 0.003 %).

Le filtrage mutualisé d'un flot de cinq individus, ayant des profils différents, a montré une légère dégradation de l'efficacité de filtrage, restant toutefois peu significative.

Ces expérimentations ont le mérite d'avoir été faites avec des données réels collectés sur une période assez longue, mais avec dans un périmètre limité. Idéalement, il serait nécessaire de les valider avec une population plus large et incluant d'autres langues. Malheureusement, il n'existe pas des ensembles de données, publiques et aussi réalistes comme ceux que nous avons utilisé, permettant d'élargir le périmètre d'expérimentation.

Ces résultats nous font estimer que le classement de messages électroniques n'est pas un problème difficile. Il est très facile d'obtenir de très bons résultats avec des méthodes assez simples. Ce qui peut le rendre un problème difficile est plutôt le niveau d'exigence des utilisateurs de la messagerie électronique : le *filtrage parfait*.

Pour qu'un *filtrage parfait* puisse être un objectif réaliste il fallait que les classes soient séparables, se qui ne semble pas être la cas, à cause des *messages gris*. Nous avons observé, dans le Chapitre 11, qu'une partie importante des erreurs de classement étaient dûs aux messages dont le jugement varie selon le destinataire.

Un deuxième point à retenir, *et c'est probablement le plus important*, concerne la variabilité des données, par rapport au niveau d'efficacité déjà atteint. Nous avons constaté, dans le Chapitre 11, l'importance de l'impact du passage des données de validation synthétiques vers des données réels sur les indicateurs d'efficacité. Compte tenu de cette variabilité des données et des niveaux d'efficacité déjà atteints, il ne semble pas que les petites améliorations d'efficacité soient significatives dans un contexte général de filtrage de *spam*.

15.2 Perspectives

15.2.1 Déploiement

L'architecture de classement que nous avons proposé et expérimenté ici nous semble la plus adaptée dans un contexte de mutualisation d'un outil de classement de messages. Elle atteint, dans notre contexte d'évaluation, un niveau d'efficacité de classement comparable à ceux de l'état de l'art (TREC 2007).

Les deux aspects de l'apprentissage : *en ligne* et *actif* répondent à des déficiences majeures des solutions de filtrage actuelles : il n'y a pas de gestion des les délais de mise à jour (perte de références temporelles) et la pertinence des retours d'information est laissée entièrement à l'initiative du destinataire et non pas du classificateur. Ces déficiences ont été décrites dans le Chapitre 9.

Cette architecture est, au moment de la présentation de cette thèse, en fonctionnement dans un certain nombre d'établissements d'enseignement et recherche français : l'Ecole des Mines de Paris, le Comité Réseau des Universités et l'Université de la Méditerranée.

Néanmoins, dans les versions en fonctionnement dans ces organismes, la boucle de retour d'information n'est pas encore automatisée. L'automatisation complète de la boucle dépend d'une fonctionnalité inexistante, actuellement, dans les logiciels de messagerie des utilisateurs, leur permettant de retourner la bonne information. Cela peut-etre réalisée sous la forme d'une *extension* (*plugin*) pour des logiciels du type *Thunderbird*.

15.2.2 Les améliorations de l'algorithme d'apprentissage

Nous avons démontré qu'il était possible d'obtenir des résultats de filtrage très bons avec un algorithme de classement très simple. Néanmoins un point reste insatisfaisant dans notre proposition, et c'est le caractère statique de deux paramètres essentiels : la marge d'apprentissage actif et la valeur résiduelle de la vitesse d'apprentissage (valeur utilisée pour l'approximation stochastique).

Nous avons vu, dans le Chapitre 11 qu'il existe un valeur optimale de marge en fonction du retard moyen dans la boucle de retour d'information et du niveau de bruit, des valeurs qui ne sont pas connues à priori. Il serait intéressant de pouvoir, à partir d'une valeur initiale arbitraire, pouvoir rechercher le point optimal de fonctionnement.

Rendre adaptative la valeur résiduelle de la vitesse d'apprentissage a l ?intérêt de minimiser le temps de réponse du classificateur à des changements brusques dans les caractéristiques des flots.

15.2.3 Chapitres "réflexions à approfondir"

Une deuxième suite de cette thèse concerne les trois derniers chapitres, qui n'ont été qu'effleurés, mais indiquent des voies qui vont dans le sens d'augmentation de la connaissance de la problématique de filtrage de spam.

15.2.4 Validation plus large

Enfin, il reste la validation des résultats obtenus, dans un périmètre plus large. Cette voie, intéressante et nécessaire, heurte sur une barrière qui est la difficulté de collecte de données réels. La solution utilisée dans cette thèse a été de dupliquer le flot de messages des individus en question et de leur demander, à la fin, de classer l'ensemble des messages. La mise en place de ce type de procédure, sur une durée assez longue et avec un nombre plus important d'individus, semble peu réaliste.

Nos expérimentations ont porté sur des ensembles de messages légitimes recueillis en France, avec des messages reçus en plusieurs langues. Nous avons remarqué que la langue prédominante dans les spams reste l'anglais. La validation de ces résultats dans d'autres langues reste à faire.

15.3 Que faut-il retenir ?

L'apport le plus important de cette thèse réside dans une meilleure connaissance et démystification de la problématique du filtrage de *spam*. Au contraire de ce que l'on pense, il ne s'agit pas d'un problème difficile : ce qui le rend difficile est la recherche d'un filtrage parfait, objectif qui semble inatteignable, puisque les classes ne semblent pas être complètement séparables.

En ce qui concerne l'architecture d'un système de filtrage, les apports les plus importants semblent être la démonstration de l ?intérêt de l'apprentissage actif en ligne dans le contexte de filtrage de spam :

- en ce qui concerne l'efficacité de classement, il est possible d'obtenir un filtrage de qualité, avec des classificateurs assez simple ;
- le retour d'information (apprentissage actif plus erreurs), en ligne rend possible l'évaluation du qualité de filtrage, ce qui n'est pas envisageable dans le contexte habituel en boucle ouverte.

Le taux d'erreur résiduel que nous avons constaté dans nos résultats semble résulter plutôt des données que d'une insuffisance des algorithmes de classement et apprentissage. Vouloir améliorer encore l'efficacité de filtrage implique forcément l'utilisation d'algorithmes de classement et d'apprentissage plus complexes. Néanmoins, vu le niveau d'efficacité déjà atteint, et les incertitudes liées aux différentes configurations de données, on peut imaginer que d'autres algorithmes plus

astucieux et aussi simple puissent apparaitre, mais il ne faut pas s'attendre à des améliorations surprenantes dans l'efficacité de classement.

La remarque de David Hand [125] faite dans un contexte général d'étude d'algorithmes d'apprentissage et commentée dans l'introduction de cette thèse nous semble plus que pertinente dans le contexte particulier de filtrage de spam.

Enfin, il ne faut pas perdre de vue que la problématique de filtrage de spam se situe dans un contexte de constante évolution technologique et "course aux armements" entre ceux qui veulent que leurs publicités arrivent dans les boites aux lettres et ceux qui ne veulent pas les recevoir. Peut-être que les conclusions de d'une étude similaire à celui-ci, et réalisée dans quelques années, seraient différentes.

Septième partie

Annexes

ANNEXE A

Probabilités et Statistique

Ce chapitre présente, très brièvement, quelques définitions ou résultats utilisés dans cette thèse.

A.1 Définitions

$X \sim P$ - une variable aléatoire X de distribution P

Définition A.1. Convergence - Soient $X_1, X_2, \ldots, X_n$ une séquence de variables aléatoires et X une autre variable aléatoire [260].

– **en Probabilité :** X_n converge vers X en probabilité ($X_n \xrightarrow{P} X$) si

$$\forall \epsilon > 0,\ P(|X - X_n| > \epsilon) \to 0, \text{ quand } n \to \infty \tag{A.1}$$

– **en Distribution :** ($X_n \rightsquigarrow X$)

$$\lim_{n\to\infty} F_n(t) = F(t) \tag{A.2}$$

– **en Moyenne Quadratique :** ou convergence en L_2, ($X_n \xrightarrow{L_2} X$)

$$\mathbf{E}[(X_n - X)^2] \to 0, \text{ quand } n \to \infty \tag{A.3}$$

A.2 Moments d'une distribution de probabilités

Le n-ième moment d'une distribution de probabilités est défini par :

$$\phi_n(X) = E\left[X^n\right] \tag{A.4}$$

Les moments courants sont celui de premier ordre et ceux d'ordre supérieure centrés sur la moyenne :

Moment	**Définition**	**Observations**
Moyenne	$\mu = E[X]$	
Variance	$\sigma^2 = V[X] = E\left[(X-\mu)^2\right]$	
Skewness	$\gamma_1 = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right]$	Asymétrie
Kurtosis	$\gamma_2 = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] - 3$	Coefficient d'Aplatissement
Covariance	$Cov[X,Y] = E[(X-\mu_X)(Y-\mu_Y)]$	
Coefficient de Corrélation	$\rho_{X,Y} = \frac{1}{\rho_X\ \rho_Y} E[(X-\mu_X)(Y-\mu_Y)]$	

Cette définition de *Kurtosis* est souvent appelée *Kurtosis en excès*, à cause de la valeur 3 déduite, permettant une comparaison avec une distribution normale.

A.3 Estimation de probabilité

L'estimation de la probabilité empirique est un problème qui arrive souvent dans, *e.g.* le classificateur bayésien naïf. Étant donné un ensemble de catégories (*e.g.* un vocabulaire), il s'agit d'estimer la probabilité empirique de chacune de ces catégories dans un échantillon (*e.g.* la probabilité empirique de chaque mot dans un corpus). L'estimateur le plus simple, celui de vraisemblance maximale, donne des résultats peu satisfaisants dans des ensembles épars (ou événements rares - les mots peu fréquents) [183, p. 198] que l'on trouve typiquement dans les applications de traitement de textes. Nous citons deux estimateurs alternatifs : Laplace et Lidstone, souvent utilisés en NLP [183, Section 6.2]

Considérons la notation suivante : $\mathcal{V} = \{m_1, \ldots\}$ est un vocabulaire de dimension $|\mathcal{V}|$ et $\mathcal{C}$ un corpus de taille $|\mathcal{C}|$: $|\mathcal{C}|$ occurrences de termes de $\mathcal{V}$. C_k est le nombre d'occurrences de m_k dans $\mathcal{C}$.

1. **Estimateur de Vraisemblance Maximale** - (MLE - Maximum Likelihood Estimation) -
$$P(m_k) = \frac{C_k}{|\mathcal{C}|} \tag{A.5}$$
2. **Estimateur de Laplace** - ou *"plus un"* (*add one*)
$$P(m_k) = \frac{C_k + 1}{|\mathcal{C}| + |\mathcal{V}|} \tag{A.6}$$
3. **Estimateur de Lidstone** - $0 < \lambda < 1$.
$$P(m_k) = \frac{C_k + \lambda}{|\mathcal{C}| + \lambda\,|\mathcal{V}|} \tag{A.7}$$

Le cas particulier $\lambda = 1/2$ correspond à l'estimateur de *Jeffreys-Perks*.

A.4 Modélisation Statistique et Critères d'Information

Deux critères d'information souvent utilisés en modélisation statistique pour choisir le modèle sont AIC (Akaike Information Criterion) et BIC (Bayes Information Criterion). Globalement, ces critères sont fondés sur la valeur de la vraisemblance maximale pénalisée par la dimension du modèle. L'idée est de, à vraisemblance égale, le modèle choisi sera celui qui est le plus simple.

Définition A.2. *AIC (Akaike Information Criterion* - Le critère *AIC* d'un modèle est défini par :

$$AIC = -2\ln(L) + 2\,k \tag{A.8}$$

avec :
- L : la valeur de la fonction de vraisemblance du modèle ;
- k : la dimension du modèle.

Définition A.3. *BIC (Bayes Information Criterion)* - Le critère *BIC* d'un modèle est défini par :

$$BIC = -2\ln(L) + k\ln(n) \tag{A.9}$$

avec :
- L : la valeur de la fonction de vraisemblance du modèle ;
- k : la dimension du modèle ;
- n : la taille équivalente de l'échantillon utilisé pour construire le modèle.

A.5 Notes Bibliographiques

Ce chapitre énumère quelques résultats utilisés dans cette thèse. On trouvera un traitement plus complet dans des textes tels Wasserman [260] (plus qu'un aide-mémoire) ou [75] (pour des explications pratiques avec le logiciel statistique *R*).

Manning et Schuze [183, Section 6.2] traitent l'utilisation des différents types d'estimateurs de probabilité dans les applications de traitement de textes en langage naturel. Church et Gale [51] [112] proposent et étudient l'utilisation de deux estimateurs (*Enhanced Good-Turing* et *Cat-Cal*) pour l'estimation de probabilités de digrammes. Zhenmei et Cercone [121] étudient l'influence des estimateurs de Laplace et Good-Turing dans le classificateur Bayésien Naïf.

Les critères d'information utilisés en modélisation statistique que nous présentons, et aussi d'autres, sont décrits en détail dans des textes tels Konish et Kitagawa [154] dédié à la modélisation statistique, [211, chap 7], [126] ou [23], ces deux derniers dédiés plutôt à l'apprentissage artificielle.

ANNEXE B

Processus Stochastiques et Séries Temporelles

Cet annexe contient un très bref résumé des outils d'analyse des séries temporelles. On trouvera une présentation plus complète dans les références mentionnées dans les notes bibliographiques.

Définition B.1. L'autocovariance d'un processus stochastique est la covariance du processus évaluée à des instants différents :

$$K_{XX}(t,s) = Cov[X_t, X_s] = E[(X_t - \mu_t)(X_s - \mu_s)] = E[X_t X_s] - \mu_t \mu_s \tag{B.1}$$

B.1 Stationnarité

Définition B.2. Soit un processus stochastique temporel X, à des valeurs réelles et en temps discret.

Si $f(X_1, X_2, \ldots, X_t)$ est une fonction mesurable (densité de probabilité conjointe, distribution de probabilité jointe, ...), alors le processus X est dit *stationnaire dans le sens strict* (*stationnarité forte*) *ssi*

$$f(X_1, X_2, \ldots, X_t) = f(X_{1+k}, X_{2+k}, \ldots, X_{t+k}) \ , \forall k \tag{B.2}$$

que l'on interprète, *in fine*, comme l'invariabilité des caractéristiques statistiques du processus, quel que soit l'instant d'observation. Une conséquence de cette définition est que, les moments de la distribution de probabilité, quel que soit l'ordre, ne dépendent pas de l'instant.

Une définition moins forte est la *stationnarité dans le sens large* (ou *stationnarité faible*, ou de *deuxième ordre*), où la contrainte ne s'applique que jusqu'aux moments de deuxième ordre :

$$\begin{aligned} E[X_i] &= \mu \ , \forall i \\ Var[X_i] &= \sigma^2 < \infty \mu \ , \ , \forall i \\ Cov[X_i, X_{i+k}] &= \rho(k) \ , \forall i, k \end{aligned} \tag{B.3}$$

B.2 Outils

B.2.1 Corrélogramme d'une Série Temporelle

Le corrélogramme d'une série temporelle stationnaire de longueur N est une estimation de sa covariance :

$$\hat{\gamma}(k) = \frac{1}{N-k} \sum_{i=0}^{N-k-1} (X(i) - \mu)(X(i+k) - \mu) \tag{B.4}$$

A noter que le nombre de termes pris en compte diminue au fur et à mesure que k augmente. D'habitude le corrélogramme est évalué pour des valeurs de k jusqu'à un tiers de la longueur de la série.

B.2.2 Variogramme

Le *demi-variogramme* d'un processus stationnaire de deuxième ordre (moyenne et variance constantes) est défini par [76, p. 40] :

$$2\,\gamma(k) = E_x\left[(Z(x) - Z(x+k))^2\right] \tag{B.5}$$

Par rapport à la fonction auto-covariance, le demi-variogramme a l'avantage de ne pas dépendre de la moyenne et, par conséquent, est défini même pour un processus non stationnaire. Bien entendu, dans ce cas, son utilisation et interprétation doit être faite avec précaution. La valeur asymptotique du demi-variogramme quand le pas tend vers infini est la variance de la série.

Remarque B.1. Pour les besoins de cette thèse, vu que nous nous intéressons plutôt à une analyse plutôt qualitative et dans le but de pouvoir représenter graphiquement les différents résultats dans une même échelle, nous utilisons le *demi-variogramme* normalisée par la variance.

Ainsi, le demi-variogramme est estimé par : que sera estimé selon :

$$\hat{\Gamma}(k) = \frac{1}{2(N-k)\sigma^2} \sum_{x=0}^{N-k-1} \left[(Z(x) - Z(x+k))^2\right] \tag{B.6}$$

De façon analogue, le demi-variogramme croisé de deux processus Y et Z, est défini par :

$$2\,\gamma_{YZ}(k) = E_x\left[(Y(x) - Y(x+k))\,(Z(x) - Z(x+k))\right] \tag{B.7}$$

Pour un traitement plus complet des demi-variogrammes, consulter, par exemple, [76] et [49].

B.2.3 Periodogramme

Pour une série stationnaire de deuxième ordre, la auto-covariance et sa Transformée de Fourier Discrète (DFT) sont définies par :

$$\begin{aligned} \rho(h) &= E_x\left[(Z(x) - \mu) \cdot (Z(x+h) - \mu)\right] \\ S(k) &= \sum_{h=0}^{N-1} \rho(h) \cdot e^{-i\,\frac{2\pi\,h}{N}\,k},\ k = 0, \ldots, N-1 \end{aligned} \tag{B.8}$$

Remarque B.2. La Transformée de Fourier Discrète de la fonction d'auto-covariance d'une processus est parfois appelée *Densité Spectrale de Puissance* à cause de son interprétation dans le domaine des télécommunications.

Le Périodogramme est un estimateur de la DFT de la fonction d'auto-covariance de la série. Il est, le plus souvent, lissé après estimation. Pour les besoins de cette thèse, nous avons utilisé le filtre le plus simple (mais pas forcément le plus efficace), une fenêtre triangulaire de Bartlett :

$$\widehat{S}'(k) = \sum_{i=-M}^{M} \frac{M+1-|i|}{(M+1)^2} \hat{S}(k+i) \ , \quad t = 0, \ldots, N-1 \tag{B.9}$$

Une étude plus complète sur les différentes méthodes de filtrage des périodogrammes se trouve dans [48, Section 7.4]

B.3 Modèles de Séries Temporelles

Définition B.3. Un processus $\{Z_t\}$ est dit purement aléatoire s'il consiste d'une suite de variables aléatoires indépendantes et également distribuées. La condition d'indépendance signifie que l'auto-covariance est nulle pour des pas différents de 0. Si la distribution des variables est normale $N(0, \sigma^2)$, ce processus est parfois appelé *bruit blanc* (white noise) $WN(0, \sigma^2)$.

B.3.1 Modèles Linéaires

Une des méthode d'étude des séries temporelles, stationnaires, consiste à considérer qu'il s'agit de la sortie d'un système dynamique ayant comme entrée un bruit blanc (réalisation d'un processus aléatoire stationnaire non corrélé de moyenne nulle : $WN(0\sigma^2)$). Nous nous intéressons ici uniquement aux modèles linéaires. Les trois modèles linéaires de base sont : AR (AutoRegressive) - $d, q = 0$, MA (Moving Average) - $p, d = 0$, $ARMA$ (AutoRegressive - Moving Average)

Définition B.4. MA(q) Moving Average : Un processus stationnaire $\{X_t\}$ est dit être un processus *MA(q)* (Moving Average d'ordre q), si $\{Z_t\}$ est un processus purement aléatoire et si $\{X_t\}$ peut être écrit sous la forme :

$$X_t = Z_t + \sum_{i=1}^{q} \theta_i Z_{t-i} \tag{B.10}$$

Définition B.5. AR(p) Auto Regressive : Un processus stationnaire $\{X_t\}$ est dit être un processus *AR(p)* (Auto Regressive d'ordre p), si $\{Z_t\}$ est un processus purement aléatoire et si $\{X_t\}$ peut être écrit sous la forme :

$$X_t - \sum_{i=1}^{p} \phi_i X_{t-i} = Z_t \tag{B.11}$$

Remarque B.3. Si $\{X_t\}$ est *AR(1)* et définit par :

$$X_t = X_{t-1} + Z_t \tag{B.12}$$

alors, on dit que $\{X_t\}$ est un processus de *marche aléatoire* (*random walk*).

Définition B.6. ARMA(p,q) Auto Regressive/Moving Average : Un processus stationnaire $\{X_t\}$ est dit être un processus *ARMA(p,q)*, si $\{Z_t\}$ est un processus purement aléatoire et si $\{X_t\}$ peut être écrit sous la forme :

$$X_t - \sum_{i=1}^{p} \phi_i X_{t-i} = Z_t + \sum_{i=1}^{q} \theta_i Z_{t-i} \tag{B.13}$$

Définition B.7. Opérateur Retard : L'opérateur L retard (ou décalage) est défini selon :

$$L \ X_t = X_{t-1} \tag{B.14}$$

B.3.2 Modèles Non Stationnaires

Processus non stationnaires : Une classe de processus non stationnaires peuvent devenir stationnaires par différentiation. C'est le cas des processus *ARIMA*, définis comme suit.

Définition B.8. ARIMA(p,d,q) Auto Regressive Integrated Moving Average : Un processus $\{X_t\}$ est un processus *ARIMA(p,d,q)*, si

$$Y_t = (1 - L)^d X_t \tag{B.15}$$

est un processus *ARMA(p,q)*.

Les processus ARIMA prennent une part

B.3.3 Modèles avec Saisonnalité

Définition B.9. Processus SARIMA(p,d,q)(P,D,Q) [38, p. 203] Un processus $\{X_t\}$ est dit $SARIMA(p,d,q)(P,D,Q)_s$ de période s si le processus $\{Y_t\}$ défini par

$$Y_t = (1 - L)^d (1 - L^s)^D \; X_t \tag{B.16}$$

est un processus $ARMA$ défini par :

$$\phi(L) \; \Phi(L^s) \; Y_t = \theta(L) \; \Theta(L^s) \; Z_t \tag{B.17}$$

$\phi(z)$, $\Phi(z)$, $\theta(z)$ et $\Theta(z)$ sont des polynômes définis par :

$$\begin{aligned} \phi(z) &= 1 - \phi_1 z - \cdots - \phi_p z^p \\ \Phi(z) &= 1 - \Phi_1 z - \cdots - \Phi_P z^P \\ \theta(z) &= 1 + \theta_1 z + \cdots + \theta_q z^q \\ \Theta(z) &= 1 + \Theta_1 z + \cdots + \Theta_Q z^Q \end{aligned} \tag{B.18}$$

et l'ensemble de paramètres (p, d, q, P, D et Q), des entiers non négatifs, définissent l'ordre du modèle. Z_t est un bruit blanc (processus stationnaire et non corrélé) $WN(0, \sigma^2)$.

B.3.4 Ajustement de Modèles

Dans nos travaux, l'ajustement du modèle associé à chaque série a été obtenu à l'aide du logiciel statistique *R* [205]. La fonction `arima` de *R* ajuste, pour un 6-tuple, définissant l'ordre du modèle, les quatre polynômes du modèle dynamique *SARIMA* correspondant - voir Equation (B.18) - les plus vraisemblables pour cet ordre et retourne la valeur de la vraisemblance de l'ajustement. Un balayage exhaustif de l'ensemble des 6-tuples possibles permet de choisir le modèle ayant la valeur de *BIC* le plus faible. Finalement, pour valider ce choix, il faut encore vérifier que la série Z_t résiduelle est bien stationnaire et non corrélée. Cette étape a été faite visuellement : la série résiduelle brute permet de constater que la moyenne et variance sont approximativement constantes et la autocorrélation (fonction `acf`) doit être "négligeable" pour toute valeur de pas non nulle. On entend par "négligeables" les valeurs d'autocorrélation dont la valeur absolue est inférieure à $1.96/\sqrt{N}$, intervalle de confiance à 95 % pour une valeur nulle, où N est le nombre de termes de la série [38] [74][1].

Cette démarche est similaire à celle proposée par Cowpertwait [74, p. 144], à l'exception que, comme méthode d'ajustement nous avons utilisé la vraisemblance maximale (*ML*) au lieu de la somme quadratique conditionnelle (*CSS*) et que comme critère de sélection du modèle nous avons utilisé le *BIC* au lieu de *AIC*[2].

[1] Ceci correspond au quantile 0.975 d'une distribution normale standard
[2] *BIC* - Bayesian Information Criteria et *AIC* - Akaike Information Criteria

B.4 Notes Bibliographiques

La bibliographie concernant les séries temporelles est assez vaste. Les livres probablement classiques sont Chatfield [48],Hamilton [124], Brockwell et Davis [38] et Lindsey [176]. L'introduction des séries temporelles par Durbin [95] est assez intéressante, même si le livre est plutôt dédié à approche par espaces d'état, que nous n'avons pas utilisé dans nos travaux. Une approche pratique de l'étude de séries temporelles à l'aide du logiciel statistique R est Cowpertwait [74].

Les séries spatiales et, en particulier les variogrammes, sont présentées dans Chiles et Delfiner [49] ou Cressie [76].

ANNEXE C

Approximation stochastique

Les techniques dites d'*approximation stochastique* (ou descente de gradient stochastique) dérivent de la méthode proposée par Robbins et Monro [210] pour la solution d'équations de la forme $M(x) = \alpha$, où $M(x)$ est une fonction monotone inconnue et dont on ne dispose que d'un nombre, possiblement infini, d'échantillons bruités $\{x_k, Y(x_k)\}$, avec $E[Y(x)] = M(x)$. Si θ est une racine de cette équation, alors la séquence x_n telle que :

$$x_{k+1} = x_k + a_k(\alpha - Y(x_k)) \tag{C.1}$$

converge vers θ en L2 et, par conséquent, en probabilité, sous certaines conditions de régularité ($Y(x)$ est uniformément bornée, $M(x)$ est non décroissante, $M'(\theta)$ existe et est positive) et la séquence des a_k satisfait :

$$a_i > 0 \ , \ \lim_{i\to\infty} a_i = 0 \ , \ \sum_{i=1}^{\infty} a_i = \infty \ \text{ et } \ \sum_{i=1}^{\infty} a_i^2 < \infty \tag{C.2}$$

$a_{k+1} = f(k, a_k)$ est une fonction quelconque satisfaisant l'Équation (C.2), *e.g.* :

$$a_i = f(i, a_{i-1}) = \frac{c}{i} \ , \ c > 0 \tag{C.3}$$

Kiefer et Wolfowitz [145] se sont inspirés de la méthode de Robbins et Monro pour proposer une méthode stochastique de recherche de la valeur extrême (minimum ou maximum) d'une fonction (convexe ou concave). Lors de chaque itération, Kiefer et Wolfowitz utilisent deux échantillons, à chaque itération, pour estimer le gradient de la fonction à minimiser (maximiser).

$$z_{k+1} = z_k + a_k \frac{(y_{2k} - y_{2k-1})}{c_k} \tag{C.4}$$

La condition de monotonicité de la fonction en étude est remplacée par une condition de convexité (ou concavité). Les séquences $\{a_k\}$ et $\{c_k\}$ devant satisfaire :

$$a_i > 0 \ , \ \lim_{i\to\infty} a_i = 0 \ , \ \lim_{i\to\infty} c_i = 0 \ , \ \sum_{i=1}^{\infty} a_i = \infty \ \text{ et } \ \sum_{i=1}^{\infty} a_i c_i < \infty \ \text{ et } \ \sum_{i=1}^{\infty} a_i^2 c_i^{-2} < \infty \tag{C.5}$$

Les méthodes d'approximation stochastique ont été largement utilisées en automatique, électronique et télécommunications pour construire des systèmes adaptatifs (voir, *e.g.* , Benveniste et all [18], devenu un classique).

Des méthodes d'optimisation convexe fondées sur l'approximation stochastique peuvent être utilisées pour l'apprentissage artificielle - voir, *e.g.* , la thèse de Léon Bottou [30].

Dans le cas d'apprentissage artificielle, il s'agit le plus souvent de rechercher la meilleure hypothèse associée à un algorithme - *e.g.* , le vecteur de poids $\mathbf{w}$ de l'algorithme discriminant linéaire que nous avons utilisé dans nos travaux, cf. Equation (9.1). Ces méthodes résultent en algorithme du type :

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \eta_k \, \nabla_{\mathbf{w_k}} C(\hat{y_k}, y_k) \tag{C.6}$$

où $C(\hat{y_k}, y_k)$ est une fonction de coût à minimiser, tel l'erreur quadratique et η_k est une séquence satisfaisant les conditions de l'Équation C.2 pour la suite $\{a_k\}$.

$$\lim_{i \to \infty} \eta_i = 0 \ , \quad \sum_i \eta_i = \infty \ , \quad \sum_i \eta_i^2 < \infty \tag{C.7}$$

Lorsqu'il s'agit d'un contexte d'apprentissage en ligne d'un processus non stationnaire, il convient de modifier les conditions imposées à la séquence η_k, de façon à avoir une valeur résiduelle positive et rendre l'algorithme adaptatif. La valeur à choisir pour cette valeur résiduelle résulte d'un compromis entre la sensibilité au bruit et la vitesse d'adaptation (poursuite) - c'est l'inertie de l'algorithme. Benveniste et all [18] suggère des valeurs de l'ordre d'un millième, comme règle pratique. La séquence $\{\eta_k\}$ peut prendre une forme du type :

$$\eta_k = \frac{\eta_0(1 - \eta_\infty)}{k} + \eta_\infty, \ \eta_0 > 0, \ 0 \leq \eta_\infty < 1 \tag{C.8}$$

Pour une étude plus approfondie, voir *e.g.* , [159, chap 11].

La bibliographie concernant les approximations stochastiques est assez vaste. Pour les aspects théoriques, les références classiques sont Benveniste et all [18], Kushner et Yin [159] ou Duflo [93].

La bibliographie concernant des applications de l'approximation stochastique aux applications d'apprentissage artificiel est moins vaste : les algorithmes de ce domaine ne sont pas tous adaptés à cela. Voir, *e.g.* [30], [165], [166], [31], [219], [108], [32], [33].

ANNEXE D

Métriques d'évaluation d'un classificateur de messages électroniques

D.1 Tables de contingence et indicateurs dérivés

Nombreux sont les indicateurs d'efficacité utilisés dans l'évaluation des applications de classement. Il arrive que des domaines d'applications différents utilisent un meme indicateur, mais en le nommant différent, juste pour des questions historiques, mais il arrive aussi souvent que des domaines différents utilisent des indicateurs différents, avec des interprétations spécifiques au domaine. Néanmoins, la plupart de ces indicateurs sont construits à partir des tables de contingence.

		Réalité	
		Ham	**Spam**
Classificateur	**Ham**	VN	FN
	Spam	FP	VP

TAB. D.1: Une table de contingence ventile les résultats d'un classement obtenus en fonction des résultats attendus.

Une table de contingence (ou table de co-occurrence) est un outil souvent utilisé lorsqu'on souhaite étudier les relations entre deux variables pouvant prendre des valeurs discrètes (ou des catégories) et présenté comme dans la Table D.1, pour le cas spécifique du classement de messages électroniques. Dans notre cas, les variables sont, dans les colonnes, le classement réel (*"Gold Standard"*) et dans, les lignes, le résultat du classificateur. La somme de chaque colonne donne le nombre réel d'éléments dans chaque classe et celle de chaque ligne donne le nombre d'éléments dans chaque classe, vus par le classificateur. Le contenu des célules est :

- **VN** : vrais négatifs - *hams* classés correctement ;
- **FN** : faux négatifs - *spams* classés en erreur ;
- **VP** : vrais positifs - *spams* classés correctement ;
- **FP** : faux positifs - *hams* classés en erreur.

Les différents rapports que l'on peut extraire de la table permettent de définir des critères d'efficacité, plus ou moins pertinents selon le type d'application. La Table D.2 présente quelque

uns.

Indicateur	**Définition**	**Remarques**
Précision (*Precision*)	$\frac{VP}{VP+FP}$	Indicateur dépendant des probabilités à priori des classes
Rappel (*Recall*)	$\frac{VP}{VP+FN}$	$(1-TFN)$
Taux de faux positifs (TFP)	$\frac{FP}{VN+FP}$	
Taux de faux négatifs (TFN)	$\frac{FN}{VP+FN}$	
Sensibilité (*Sensibility*, *Power*)	$\frac{VP}{VP+FN}$	$(1-TFN)$
Spécificité (*Specificity*)	$\frac{VN}{VN+FP}$	$(1-TFP)$
Exactitude (*Accuracy*)	$\frac{VP+VN}{VP+FP+VN+FN}$	Indicateur dépendant des probabilités à priori des classes
Taux d'erreur global	$\frac{FP+FN}{VP+FP+VN+FN}$	$(1-Exactitude)$

Tab. D.2: Quelques indicateurs définis à partir d'une table de contingence

La pertinence de chaque indicateur dépende de l'application où il sera utilisé. Dans une application de classement de messages électroniques, la probabilité à priori des classes varie non seulement dans le temps, mais aussi d'individu à individu. Plus généralement, les indicateurs dont l'évaluation utilise des valeurs pris dans plusieurs colonnes dépendent de la probabilité à priori des classes et ne sont pas adéquats. Le couple *Précision* et *Rappel*, souvent utilisé dans les applications de recherche documentaire, est un exemple d'indicateur à éviter, pour cette raison, dans les applications de classement de messages électroniques.

Dans un flot de messages électroniques, les classes sont asymétriques, non seulement dans leur répartition, mais aussi dans le coût des erreurs de classement. Ainsi, des propositions ont été faites de, par exemple, pondération des erreurs par un coefficient de coût associé à chaque classe [10], mais s'il est vrai que les coûts sont asymétriques, non seulement il n'y a pas de critère objectif permettant leur évaluation [128] mais aussi le coût varie selon le genre de message, à l'intérieur de chaque classe. Il semble plus utile de renseigner le taux d'erreur par classe et laisser le jugement à la discrétion du destinataire des messages.

D.2 *ROC* (Receiver Operating Characteristic)

Les indicateurs dérivés d'une table de contingence ont l'inconvénient d'être spécifiques à un point d'opération particulier - une valeur de seuil - et ne disent rien sur l'efficacité du filtre à d'autres points d'opération.

Certains classificateurs présentent présentent uniquement un résultat binaire - *ham/spam* - (*hard classifiers*) tandis que d'autres indiquent le résultat sous la forme d'un score avec valeur numérique ("soft classifiers").

Pour un soft classifier, le *ROC* [101] est représenté graphiquement par une courbe présentant le *Taux de Vrai Positifs* en fonction du *Taux de Faux Positifs*, paramétrée par le seuil de classement, pour un *soft classifier*, *e.g.* la Figure D.1.

Les courbes *ROC* ont des caractéristiques visuelles que l'on peut interpréter facilement :
- la courbe *ROC* ne dépend pas de la répartition des classes puisque ;
- la courbe *ROC* est entièrement comprise dans le carré de sommets opposés (0,0) et (1,1) : les variables représentées sont des probabilités ;
- un classificateur idéal (au cas où il existerait un) ne commet pas d'erreurs et donc sa courbe *ROC* se confond avec les côtés gauche et supérieur de ce carré ;

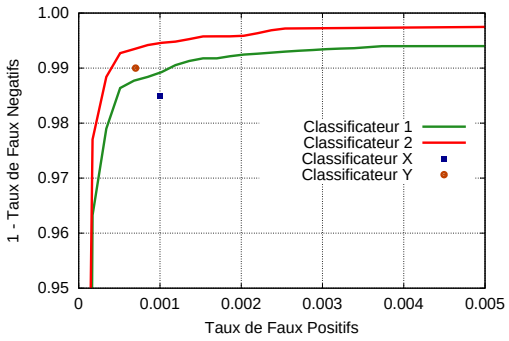


FIG. D.1: Exemple *ROC*

- le segment reliant les sommets (0,0) et (1,1) définit une zone d'incertitude où l'efficacité du classificateur est identique à un choix aléatoire ;
- pour comparer deux points d'opération, il suffit de sélectionner celui que est plus vers le haut et vers la gauche ;
- pour comparer deux classificateurs, le meilleur est celui dont la courbe est *ROC* est la plus proche du coin en haut et à gauche.

La courbe *ROC* a une autre propriété intéressante : la surface sous la courbe correspond à la probabilité de prendre, au hasard, un exemple de chaque classe et d'avoir dans le bon ordre les scores attribués par le classificateur. Cette valeur est souvent présentée par la sigle *AUC* (Area Under the Curve) ou *ROCA* (*ROC Area*) ou plutôt le complément à 1 : *1-AUC* ou *1-ROCA*. En effet, l'AUC correspond au résultat d'un test de rang de Wilcoxon [134, p.128].

La l'indicateur *1-AUC* représente globalement l'efficacité de classement, quelque soit la valeur de seuil choisie.

D.3 Notes Bibliographiques

Peu de références bibliographiques sont dédiées à l'évaluation d'algorithmes en apprentissage artificielle. Des références intéressantes sont les livres récents de Japkowicz et all [134], qui traite le probléme général d'évaluation d'algorithmes de classement, et Buttcher et all [40], orienté aux algorithmes de recherche documentaire.

Tom Fawcett a écrit un tutoriel intéressant concernant l'utilisation de *ROC* [101].

Gordon Cormack a mis au point, pour les conférences TREC, une méthodologie d'évaluation de filtres anti-spam [61] [63] [69].

ANNEXE E

Théorie d'Information

E.1 Définitions

Définition E.1 (Entropie)**.** Soit X une variable aléatoire discrète associée à une distribution de probabilité P en $\mathcal{X}$. L'entropie de X est définie par :

$$H(X) = -\sum_{x \in \mathcal{X}} P(x) \log P(x) \tag{E.1}$$

que l'on note aussi sous la forme $H(P)$ lorsque l'on veut mettre en valeur le fait que l'on s'intéresse à l'entropie associée à une distribution de probabilité.

L'entropie est, parmi les différentes interprétations, une mesure de l'incertitude moyenne que l'on peut avoir sur la valeur prise par la variable aléatoire X.

Définition E.2 (Entropie Jointe)**.** - Soient X et Y deux variables aléatoires dont la distribution conjointe est $P(X, Y)$. L'entropie conjointe é définie par :

$$H(X, Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x, y) \log P(x, y) \tag{E.2}$$

Définition E.3 (Entropie Conditionnelle)**.**

$$\begin{aligned} H(Y|X) &= \sum_{x \in \mathcal{X}} P(x) H(Y|X = x) \\ &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x, y) \log P(y|x) \end{aligned} \tag{E.3}$$

L'entropie conditionnelle indique l'incertitude que l'on a de la valeur de X sachant Y.

Propriétés de l'Entropie

* Non négativité :

$$H(X) \geq 0$$

* Réduction d'entropie par conditionnement :

$$H(X|Y) \leq H(X)$$

* Distribution conjointe :

$$H(X_1, X_2, \ldots, X_n) \leq \sum_{i=1}^{n} H(X_i)$$

* Maximum :

$$H(X) < |\mathcal{X}|$$

* La fonction entropie :

$$H(p) = -p \log \frac{1}{p} - (1-p) \log \frac{1}{1-p}$$

est concave en p.

Information Mutuelle

$$\begin{aligned} I(X;Y) &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x,y) \log \frac{P(x,y)}{P(x)P(y)} \\ &= x \end{aligned} \tag{E.4}$$

Entropie Relative

$$D(P \parallel Q) = E_P \left[log \frac{P(X)}{Q(X)} \right] = \sum_{x \in \mathcal{X}}^{N} P(x) \frac{P(x)}{Q(x)} \tag{E.5}$$

Règle de Chainage Entropie

$$\begin{aligned} H(X,Y) &= H(X) + H(Y|X) \\ H(X_1, X_2, \ldots, X_N) &= \sum_{i=0}^{N} H(X_i | X_1, X_2, \ldots, X_{i-1}) \end{aligned} \tag{E.6}$$

$$I(X_1, X_2, \ldots, X_N; Y) = \sum_{i=0}^{N} I(X_i; Y | X_1, X_2, \ldots, X_{i-1}) \tag{E.7}$$

$$D(P(X,Y) \parallel Q(X,Y) = D(P(X) \parallel Q(X)) + D(P(Y|X) \parallel Q(Y|X)) \tag{E.8}$$

ANNEXE F

Treillis et Algèbre de Boole

Ce chapitre est une présentation très brève des Treillis et de l'Algèbre de Boole avec seuls les concepts dont nous avons besoin. Un petit arrêt prolongé sur les ensembles se fait nécessaire pour valider la notation. Pour une présentation plus détaillée, on peut se référer, par exemple, à [22] [26] [147], des livres dont nous nous sommes inspirés pour ce contenu.

F.1 Ensembles

Nous considérons acquise la notion intuitive d'*ensemble* et nous nous limitons à la notation.

Un *ensemble* est une collection d'objets distincts. Les objets d'un ensemble sont les *éléments* ou *membres*. On note :

- $A = \{a_1, a_2, \ldots, a_n\}$, l'ensemble dont les éléments sont $a_1, a_2, \ldots, a_n$;
- $A = \{x \in A \mid P(x)\}$, l'ensemble des éléments $x \in A$ qui vérifient la propriété $P(x)$;
- $a \in A$, a est un élément de A ;
- $A = \varnothing$, A est un ensemble qui ne contient aucun élément, et on le nomme un *ensemble vide* ;

Définition F.1. Deux ensembles sont égaux s'ils contiennent exactement les mêmes éléments.

Définition F.2. Un ensemble A est un sous-ensemble d'un ensemble B, si tout élément de A est aussi un élément de B. On dit aussi que A est inclus dans B ou que A est une partie de B et on note : $A \subset B$.

Définition F.3. Union : Soient A et B deux ensembles, on dit que C est l'union de A et B si tout élément de C est un élément de A ou de B et on note

$$C = A \cup B = \{x \mid x \in A \text{ ou } x \in B\} \tag{F.1}$$

Définition F.4. Intersection : Soient A et B deux ensembles, on dit que C est l'intersection de A et B si tout élément de C est un élément de A et de B et on note

$$C = A \cap B = \{x \mid x \in A \text{ et } x \in B\} \tag{F.2}$$

Définition F.5. Deux ensembles sont *disjoints* ou *mutuellement exclusifs* s'ils n'ont aucun élément en commun, *i.e.* $A \cap B = \varnothing$.

Définition F.6. Relation binaire sur A : Soit A un ensemble. On appelle relation binaire sur A toute correspondance de $A \times A$ vers A.

F.2 Ordre Partiel et Treillis

Définition F.7. Relation d'ordre : Une relation binaire $\mathscr{R}$ sur un ensemble A est dite relation d'ordre si elle est :

1. **réflexive** : $\forall x \in A,\ x \mathscr{R} x$
2. **transitive** : $\forall x, y, z \in A : \{x \mathscr{R} y \text{ et } y \mathscr{R} z\} \Rightarrow x \mathscr{R} z$
3. **antisymétrique** : $\forall x, y \in A : \{x \mathscr{R} y \text{ et } y \mathscr{R} x\} \Rightarrow x = z$

On note souvent $x \mathscr{R} y$ par $x \leqslant y$ si x est "*inférieur ou égal*" à y, $x \geqslant y$ si x est "*supérieur ou égal*" à y, $x < y$ au lieu de $x \leqslant y$ et $x \neq y$. On note $(A, \leqslant)$ un ensemble muni d'une relation d'ordre.

Définition F.8. Ordre Total et Ordre Partiel : On dit que l'ensemble A est totalement ordonné par la relation d'ordre $\leqslant$ *ssi* cette relation est définie pour toute paire d'éléments de A, *c.à.d.* $\forall x, y \in A,\ \Rightarrow x \leqslant y$ ou $x \geqslant y$.

Un ordre est partiel quand il n'est pas total, *c.à.d.* l'ordre n'est pas défini pour certaines paires d'éléments de A et on dit que les couples pour lesquels l'ordre n'est pas défini ne sont pas comparables.

Définition F.9. Majorant et Borne Supérieure : Soit $(A, \leqslant)$ un ensemble ordonné et E un sous ensemble de A. On dit que u est un *majorant* de E *ssi* $u \in A$ et $\forall x \in E \Rightarrow x \leqslant u$. On dit que E est majoré s'il existe au moins un majorant de E.

La *borne supérieure* de l'ensemble E, si elle existe, est le plus petit des majorants de E. m est une borne supérieure de E si :

$$E \subset A, \forall x, y \in E,\ m \in A,\ x \leqslant m,\ y \leqslant m, \quad si \quad u \in A,\ x \leqslant u,\ y \leqslant u \ \Rightarrow\ m \leqslant u \tag{F.3}$$

Définition F.10. Minorant et Borne Inférieure : Soit $(A, \leqslant)$ un ensemble ordonné et E un sous ensemble de A. On dit que b est un *minorant* de E *ssi* $b \in A$ et $\forall x \in E \Rightarrow b \leqslant x$. On dit que E est minoré s'il existe au moins un minorant de E.

La *borne inférieure* de l'ensemble E, si elle existe, est le plus grand des minorants de E. l est une borne inférieure de E si :

$$E \subset A, \forall x, y \in E,\ l \in A,\ l \leqslant x,\ l \leqslant y, \quad si \quad b \in A,\ b \leqslant x,\ b \leqslant y \ \Rightarrow\ b \leqslant l \tag{F.4}$$

On remarque que les bornes supérieure et inférieure d'un ensemble sont uniques. Les bornes supérieures sont des propriétés duales l'une de l'autre.

Les définitions ci-dessus de Borne Supérieure et Inférieure font référence à des sous ensembles mais, elles prennent une toute autre importance en tant que relation binaire appliquée à des couples, si l'on restreint l'ensemble E aux ensembles de deux éléments, que l'on note :

$$x, y \in A \quad \begin{cases} l = x \wedge y & l \in A \\ m = x \vee y & m \in A \end{cases} \tag{F.5}$$

En abrégé, ces bornes sont désignées par *sup* et inf et on dit que l est le inf de (x, y) et m le *sup* de (x, y).

Définition F.11. Couverture Dans un ensemble ordonné A, on dit que y *couvre* x (ou que x est couvert par y, si $x \leqslant y$ et il n'y a aucun $a \in A$ tel que $x < a < y$. Cette succession immédiate est notée par $x \prec y$.

Le Diagramme de Hesse constitue une forme de représentation usuelle d'ensembles ordonnés. Dans ce diagramme, les éléments éléments sont représentés par des sommets et les relations de couverture (succession immédiate) sont représentées par des arêtes.

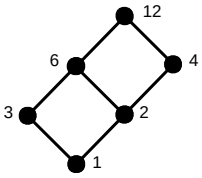


FIG. F.1: Diagramme de Hesse pour l'ensemble des diviseurs de 12, utilisant la divisibilité comme relation d'ordre.

Exemple F.1. Soit $A = \{1, 2, 3, 4, 6, 12\}$ l'ensemble des diviseurs de 12. Si l'on ordonne cet ensemble par l'ordre naturelle on obtient une chaine. Par contre, si on utilise la divisibilité comme relation d'ordre, on obtient le diagramme de la figure (F.1).

Définition F.12. Treillis : Un Treillis[1] est un ensemble ordonné $(E, \leqslant)$ dans lequel tout couple d'éléments (x, y) possède à la fois une borne supérieure et une borne inférieure. Dans un treillis, *borne supérieure* et *borne inférieure* sont des opérations binaires $\vee : E \times E \to E$ et $\wedge : E \times E \to E$.

Propriétés F.2.1. *Propriétés des Treillis : Dans un ensemble ordonné quelconque P, les opérations inf et sup satisfont aux règles suivantes :*

P1. Idempotence : $x \wedge x = x$, $x \vee x = x$
P2. Commutativité : $x \wedge y = y \wedge x$ *et* $x \vee y = y \vee x$
P3. Associativité : $x \wedge (y \wedge z) = (x \wedge y) \wedge z$ *et* $x \vee (y \vee z) = (x \vee y) \vee z$
P4. Absorption : $x \wedge (x \vee y) = x \vee (x \wedge y) = x$

Définition F.13. Treillis Modulaire : est celui qui satisfait :
P5. Si $x \leqslant y$, on a $x \vee (y \wedge z) = (x \vee y) \wedge z$

Définition F.14. Treillis Distributif : On dit qu'un treillis est distributif lorsqu'il satisfait les règles :
P6. $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
P7. $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

F.3 Algèbre de Boole

Définition F.15. Complément d'un élément : on appelle complément d'un élément x d'un treillis A contenant O et I, un élément $y \in A$ tel que $x \wedge y = O$ et $x \vee y = I$. Le treillis est dit complémenté si tous les éléments ont des compléments.

Définition F.16. Un Treillis de Boole est un treillis distributif où tout élément x a un complément unique x'. x et x' vérifient :
P8. $x \wedge x' = O$ et $x \vee x' = I$
P9. $(x')' = x$
P10. $(x \wedge x') = x' \vee y'$ et $(x \vee x') = x' \wedge y'$

Définition F.17. Une Algèbre de Boole est un ensemble A muni des deux opérations binaires $\vee$ et $\wedge$ et d'une opération unaire $'$ satisfaisant *P1* à *P10*.

[1] Certains auteurs utilisent l'expression *Ensemble Réticulé* à la place de *Treillis*.

ANNEXE G

Résultats détaillés - Chapitre 11

Bruit (%)	1-AUC (%)	Erreur (%) Ham	Erreur (%) Spam	Query (%) Ham	Query (%) Spam	Apprentissage (%) Ham	Apprentissage (%) Spam
0.0	0.005481	0.2987	0.0355	2.0949	0.5949	2.0949	0.5949
1.0	0.005523	0.3026	0.0360	2.1196	0.6061	2.1196	0.6061
2.0	0.005621	0.3130	0.0404	2.1936	0.6286	2.1923	0.6286
4.0	0.006335	0.3442	0.0429	2.2910	0.6776	2.2897	0.6770
8.0	0.007076	0.3857	0.0559	2.5456	0.7917	2.5339	0.7906
12.0	0.008613	0.4364	0.0726	2.8404	0.9537	2.8326	0.9523
16.0	0.009831	0.5455	0.0868	3.2001	1.0849	3.1858	1.0823

TAB. G.1: Impact du bruit (erreurs) dans le retour d'information d'étiquette - retour immédiat

Bruit (%)	1-AUC (%)	Erreur (%) Ham	Erreur (%) Spam	Query (%) Ham	Query (%) Spam	Apprentissage (%) Ham	Apprentissage (%) Spam
0.0	0.006785	0.3234	0.0404	2.6910	0.8773	1.9871	0.5661
1.0	0.006400	0.3390	0.0435	2.7066	0.9229	2.0312	0.5897
2.0	0.006762	0.3559	0.0453	3.0625	0.9918	2.1520	0.6373
4.0	0.007770	0.3974	0.0510	3.1053	1.0719	2.3157	0.7027
8.0	0.011843	0.4442	0.0709	3.4884	1.7622	2.5910	1.0560
12.0	0.031575	0.6922	0.4655	4.9664	6.3143	3.6066	3.4653
16.0	0.055315	0.9676	0.7024	7.5172	10.4261	5.4561	6.5262

TAB. G.2: Impact du bruit (erreurs) dans le retour d'information d'étiquette - retour aléatoire de distribution exponentielle de moyenne 6 h

Bruit (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.0	0.007785	0.3442	0.0516	3.8066	1.3855	1.8819	0.5323
1.0	0.008216	0.3585	0.0628	4.2132	1.5048	2.0520	0.5767
2.0	0.009200	0.3974	0.0643	4.2469	1.7014	2.1663	0.6586
4.0	0.012068	0.4390	0.1268	4.8872	2.5254	2.5066	0.9886
8.0	0.020070	0.6013	0.1473	6.1899	5.1171	3.5768	2.2130
12.0	0.081218	1.1974	0.5750	11.1082	17.4990	6.4678	9.0222
16.0	0.280995	1.9546	2.1314	15.9669	33.3711	10.2056	20.2501

Tab. G.3: Impact du bruit (erreurs) dans le retour d'information d'étiquette - retour aléatoire de distribution exponentielle de moyenne 24 h

Bruit (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.0	0.009455	0.3675	0.0562	3.9911	1.5898	1.8624	0.5292
1.0	0.009660	0.4052	0.0553	4.4495	1.7875	2.0585	0.6070
2.0	0.010925	0.4104	0.0663	4.5521	1.8708	2.1390	0.6877
4.0	0.011407	0.4455	0.0842	5.3872	2.4637	2.6235	0.9171
8.0	0.026360	0.6611	0.2306	7.8003	7.6511	4.0625	3.1468
12.0	0.127131	1.3858	1.2019	13.8031	26.1589	7.7341	13.4008
16.0	0.414701	2.5469	2.5784	19.6202	36.3942	11.7758	22.1371

Tab. G.4: Impact du bruit (erreurs) dans le retour d'information d'étiquette - retour aléatoire de distribution exponentielle de moyenne 48 h

Retard	1-AUC	Erreur (%)		Query (%)		Apprentissage (%)	
(h)	(%)	Ham	Spam	Ham	Spam	Ham	Spam
0	0.005481	0.2987	0.0355	2.0949	0.5949	2.0949	0.5949
6	0.006448	0.3338	0.0386	2.6079	0.7710	2.0235	0.5713
12	0.006565	0.3429	0.0435	2.7910	0.8998	1.9845	0.5591
24	0.007059	0.3546	0.0418	3.1443	1.0413	1.9546	0.5505
36	0.007240	0.3727	0.0461	3.5443	1.1520	1.9273	0.5393
48	0.007662	0.3740	0.0519	3.7521	1.2711	1.9157	0.5341
60	0.008505	0.3688	0.0527	3.9716	1.4077	1.9027	0.5303
72	0.008611	0.3624	0.0551	4.2339	1.4956	1.8988	0.5332
96	0.008119	0.3688	0.0530	3.9924	1.5446	1.8715	0.5300
120	0.008665	0.3701	0.0551	3.9534	1.5477	1.8767	0.5387
168	0.009408	0.3935	0.0597	4.2054	1.6524	1.8754	0.5249

Tab. G.5: Résultats de classement en fonction du retard de retour d'étiquette (constant). Marge d'apprentissage actif : 0.35

Retard	1-AUC	Erreur (%)		Query (%)		Apprentissage (%)	
(h)	(%)	Ham	Spam	Ham	Spam	Ham	Spam
0	0.005481	0.2987	0.0355	2.0949	0.5949	2.0949	0.5949
6	0.006785	0.3234	0.0404	2.6910	0.8773	1.9871	0.5661
12	0.006833	0.3403	0.0415	3.3066	1.0145	1.9728	0.5453
24	0.007785	0.3442	0.0516	3.8066	1.3855	1.8819	0.5323
36	0.009568	0.3818	0.0631	4.3339	1.7299	1.8650	0.5225
48	0.009455	0.3675	0.0562	3.9911	1.5898	1.8624	0.5292
60	0.008753	0.3727	0.0746	3.9027	1.7264	1.8559	0.5202
72	0.009083	0.3831	0.0588	4.2209	1.6186	1.8624	0.5231
96	0.009360	0.3831	0.0628	4.1677	1.6924	1.8364	0.5234
120	0.009575	0.4052	0.0640	4.2183	1.6613	1.8598	0.5283
168	0.009522	0.3805	0.0625	4.1326	1.7155	1.8728	0.5257

Tab. G.6: Résultats de classement en fonction du retard moyen de retour d'étiquette (aléatoire de distribution exponentielle). Marge d'apprentissage actif : 0.35

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.011662	0.3247	0.0568	0.6000	0.1401	0.6000	0.1401
0.10	0.008518	0.2870	0.0464	0.7767	0.1847	0.7767	0.1847
0.15	0.007193	0.2636	0.0409	0.9585	0.2323	0.9585	0.2323
0.20	0.006450	0.2766	0.0369	1.1325	0.2848	1.1325	0.2848
0.25	0.006129	0.2714	0.0372	1.3767	0.3542	1.3767	0.3542
0.30	0.005861	0.2844	0.0372	1.6819	0.4496	1.6819	0.4496
0.35	0.005481	0.2987	0.0355	2.0949	0.5949	2.0949	0.5949
0.40	0.005113	0.3182	0.0366	2.8222	0.8577	2.8222	0.8577
0.45	0.005025	0.3377	0.0378	4.4989	1.4881	4.4989	1.4881
0.50	0.005588	0.4299	0.0481	100.0000	100.0000	100.0000	99.9997

TAB. G.7: Impact de la marge d'apprentissage actif - Retour d'information immédiat - Niveau de Bruit = 0 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.012297	0.3481	0.0574	0.6221	0.1473	0.6221	0.1473
0.10	0.009229	0.2935	0.0464	0.7793	0.1853	0.7780	0.1853
0.15	0.007448	0.2701	0.0427	0.9572	0.2306	0.9533	0.2306
0.20	0.007581	0.2740	0.0392	1.1416	0.2859	1.1416	0.2856
0.25	0.006200	0.2727	0.0389	1.3897	0.3632	1.3897	0.3632
0.30	0.006155	0.2961	0.0360	1.7040	0.4557	1.7014	0.4557
0.35	0.005523	0.3026	0.0360	2.1196	0.6061	2.1196	0.6061
0.40	0.005190	0.3260	0.0395	2.9105	0.8840	2.9105	0.8837
0.45	0.005067	0.3429	0.0404	4.6963	1.5936	4.6963	1.5936
0.50	0.010877	0.4546	0.0481	100.0000	100.0000	100.0000	99.9997

TAB. G.8: Impact de la marge d'apprentissage actif - Retour d'information immédiat - Niveau de Bruit = 1 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.010751	0.3429	0.0657	0.6065	0.1467	0.6013	0.1458
0.10	0.007810	0.2948	0.0490	0.7948	0.1943	0.7922	0.1934
0.15	0.008073	0.2766	0.0453	0.9767	0.2404	0.9741	0.2404
0.20	0.006616	0.2831	0.0404	1.1962	0.2928	1.1923	0.2917
0.25	0.006383	0.2792	0.0418	1.4195	0.3724	1.4182	0.3721
0.30	0.005940	0.3000	0.0380	1.7118	0.4808	1.7105	0.4805
0.35	0.005621	0.3130	0.0404	2.1936	0.6286	2.1923	0.6286
0.40	0.005972	0.3390	0.0398	3.0196	0.9431	3.0196	0.9431
0.45	0.005509	0.3611	0.0441	4.9781	1.7043	4.9781	1.7043
0.50	0.016021	0.6104	0.0608	100.0000	100.0000	100.0000	99.9997

Tab. G.9: Impact de la marge d'apprentissage actif - Retour d'information immédiat - Niveau de Bruit = 2 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.013613	0.3481	0.0654	0.6364	0.1571	0.6286	0.1551
0.10	0.010157	0.2896	0.0536	0.8182	0.1951	0.8156	0.1945
0.15	0.008014	0.2974	0.0467	1.0299	0.2418	1.0247	0.2412
0.20	0.008063	0.3039	0.0458	1.2117	0.3046	1.2091	0.3032
0.25	0.007366	0.3195	0.0429	1.4988	0.3799	1.4949	0.3781
0.30	0.005570	0.3299	0.0421	1.8598	0.4831	1.8598	0.4822
0.35	0.006335	0.3442	0.0429	2.2910	0.6776	2.2897	0.6770
0.40	0.006015	0.3481	0.0429	3.1573	1.0194	3.1560	1.0189
0.45	0.006259	0.3883	0.0565	5.4483	2.0988	5.4457	2.0985
0.50	0.037899	0.9935	0.1695	100.0000	100.0000	100.0000	99.9997

Tab. G.10: Impact de la marge d'apprentissage actif - Retour d'information immédiat - Niveau de Bruit = 4 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.015706	0.3611	0.0663	0.6805	0.1571	0.6624	0.1554
0.10	0.011701	0.3766	0.0574	0.8728	0.2046	0.8637	0.2018
0.15	0.009363	0.3468	0.0539	1.0806	0.2513	1.0728	0.2496
0.20	0.008416	0.3312	0.0484	1.3169	0.3326	1.3130	0.3315
0.25	0.007889	0.3598	0.0458	1.6014	0.3992	1.5884	0.3972
0.30	0.007209	0.3572	0.0487	1.9014	0.5917	1.8975	0.5911
0.35	0.007076	0.3857	0.0559	2.5456	0.7917	2.5339	0.7906
0.40	0.007056	0.4234	0.0620	3.5729	1.2944	3.5651	1.2941
0.45	0.008575	0.4831	0.0896	7.1353	3.4137	7.1327	3.4137
0.50	0.218627	2.5897	0.8577	100.0000	100.0000	100.0000	99.9997

TAB. G.11: Impact de la marge d'apprentissage actif - Retour d'information immédiat - Niveau de Bruit = 8 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.016013	0.4312	0.0937	0.7429	0.1816	0.6961	0.1721
0.10	0.013290	0.3961	0.0755	0.9468	0.2369	0.9104	0.2317
0.15	0.011702	0.4221	0.0741	1.1897	0.3289	1.1637	0.3248
0.20	0.011803	0.4455	0.0680	1.5040	0.3914	1.4832	0.3865
0.25	0.009316	0.4701	0.0787	1.8897	0.5326	1.8780	0.5309
0.30	0.008938	0.4727	0.0778	2.2676	0.7119	2.2403	0.7099
0.35	0.009831	0.5455	0.0868	3.2001	1.0849	3.1858	1.0823
0.40	0.013703	0.6572	0.1608	5.4483	2.6456	5.4353	2.6427
0.45	0.098798	1.3572	0.9572	17.3241	25.4133	17.3137	25.4107
0.50	2.050730	7.9769	4.7533	100.0000	100.0000	100.0000	99.9997

TAB. G.12: Impact de la marge d'apprentissage actif - Retour d'information immédiat - Niveau de Bruit = 16 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.014408	0.4312	0.0896	0.8091	0.2845	0.5650	0.1323
0.10	0.011173	0.3585	0.0700	1.0234	0.3917	0.7143	0.1703
0.15	0.009054	0.3546	0.0473	1.3624	0.4041	0.8935	0.2176
0.20	0.009085	0.3039	0.0447	1.6105	0.4885	1.0974	0.2761
0.25	0.008709	0.3247	0.0412	1.8494	0.5825	1.3052	0.3364
0.30	0.006901	0.3351	0.0418	2.2637	0.6961	1.6027	0.4283
0.35	0.006785	0.3234	0.0404	2.6910	0.8773	1.9871	0.5661
0.40	0.006188	0.3598	0.0404	3.5183	1.1486	2.7053	0.8165
0.45	0.006100	0.3883	0.0438	5.4275	1.8579	4.2755	1.4074
0.50	0.006620	0.4870	0.0542	100.0000	100.0000	99.7013	99.8723

Tab. G.13: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 6 h - Niveau de Bruit = 0 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.014517	0.4870	0.0893	0.8974	0.2513	0.5844	0.1358
0.10	0.014873	0.4000	0.0651	1.1676	0.3805	0.7507	0.1830
0.15	0.011078	0.3727	0.0536	1.3468	0.4041	0.9143	0.2297
0.20	0.008500	0.3208	0.0478	1.5884	0.4580	1.1078	0.2796
0.25	0.008160	0.3338	0.0461	1.9637	0.5617	1.3546	0.3502
0.30	0.007720	0.3585	0.0441	2.4053	0.7569	1.6624	0.4433
0.35	0.006400	0.3390	0.0435	2.7066	0.9229	2.0312	0.5897
0.40	0.006666	0.3688	0.0424	3.7456	1.2350	2.8274	0.8618
0.45	0.006479	0.3870	0.0412	5.7587	2.0037	4.4963	1.5071
0.50	0.011918	0.5000	0.0657	100.0000	100.0000	99.7221	99.8792

Tab. G.14: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 6 h - Niveau de Bruit = 1 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.016900	0.5221	0.1029	0.9935	0.2882	0.6260	0.1438
0.10	0.015027	0.4052	0.0795	1.1117	0.4202	0.7611	0.1896
0.15	0.009888	0.3585	0.0669	1.3312	0.4424	0.9416	0.2384
0.20	0.008422	0.3468	0.0470	1.6429	0.4911	1.1390	0.2934
0.25	0.007612	0.3429	0.0464	1.9637	0.6208	1.3702	0.3614
0.30	0.008030	0.3312	0.0418	2.3637	0.7073	1.7092	0.4548
0.35	0.006762	0.3559	0.0453	3.0625	0.9918	2.1520	0.6373
0.40	0.006658	0.3507	0.0421	3.8833	1.2837	2.9404	0.9105
0.45	0.006416	0.4247	0.0476	6.4652	2.2720	5.1288	1.7045
0.50	0.017913	0.5896	0.0882	100.0000	100.0000	99.7221	99.8792

TAB. G.15: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 6 h - Niveau de Bruit = 2 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.019054	0.4637	0.1038	0.9208	0.2920	0.6247	0.1585
0.10	0.012336	0.4221	0.0821	1.1884	0.3655	0.7793	0.1997
0.15	0.012366	0.3870	0.0758	1.4676	0.4715	0.9884	0.2508
0.20	0.009502	0.4013	0.0651	1.8299	0.6076	1.2546	0.3170
0.25	0.009431	0.3909	0.0692	2.1624	0.8500	1.5481	0.4245
0.30	0.008658	0.3675	0.0496	2.4936	0.8358	1.8079	0.5156
0.35	0.007770	0.3974	0.0510	3.1053	1.0719	2.3157	0.7027
0.40	0.007688	0.4078	0.0551	4.3872	1.6106	3.3300	1.0699
0.45	0.008161	0.4546	0.0654	7.4951	3.8797	6.0028	2.6842
0.50	0.034010	0.9494	0.1721	100.0000	100.0000	99.7221	99.8792

TAB. G.16: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 6 h - Niveau de Bruit = 4 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.024396	0.5650	0.2283	1.0208	0.4747	0.6559	0.2150
0.10	0.015106	0.4585	0.1242	1.3026	0.4323	0.8650	0.2415
0.15	0.015494	0.4221	0.0738	1.6247	0.4476	1.1143	0.2750
0.20	0.012361	0.4779	0.1216	1.9858	0.9197	1.3481	0.4401
0.25	0.011394	0.4156	0.0819	2.1287	0.8981	1.6312	0.5073
0.30	0.012899	0.4650	0.1352	3.0897	1.8233	2.1546	0.9030
0.35	0.011843	0.4442	0.0709	3.4884	1.7622	2.5910	1.0560
0.40	0.013718	0.5299	0.0954	5.9184	3.9426	4.4002	2.3689
0.45	0.040979	0.7961	0.3165	17.4981	31.6971	13.0213	21.9319
0.50	0.164003	2.4196	0.7076	100.0000	100.0000	99.7221	99.8792

Tab. G.17: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 6 h - Niveau de Bruit = 8 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.030288	0.7065	0.2032	1.1520	0.4081	0.8013	0.2237
0.10	0.048310	0.6741	0.4983	1.5663	1.1307	1.0884	0.5214
0.15	0.024441	0.6208	0.1657	1.9871	0.9134	1.3832	0.5125
0.20	0.020141	0.6572	0.2412	2.5611	1.4766	1.8170	0.8162
0.25	0.040773	0.6909	0.5946	3.3300	3.3722	2.3546	1.7544
0.30	0.029299	0.7728	0.2827	3.7937	3.8501	2.8131	2.2726
0.35	0.055315	0.9676	0.7024	7.5172	10.4261	5.4561	6.5262
0.40	0.206590	1.6299	2.0028	18.2007	38.1452	13.7733	27.7589
0.45	0.626636	3.3664	3.5512	45.6589	74.1518	37.2716	63.7202
0.50	1.885750	8.3120	4.4965	100.0000	100.0000	99.7221	99.8792

Tab. G.18: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 6 h - Niveau de Bruit = 16 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.053748	1.0052	0.5240	2.1105	1.3503	0.5039	0.1176
0.10	0.026610	0.7169	0.1943	2.4871	1.1904	0.6585	0.1556
0.15	0.021507	0.6026	0.1378	2.8871	1.6025	0.8052	0.1969
0.20	0.014277	0.4585	0.0836	2.6663	1.1615	1.0039	0.2459
0.25	0.010668	0.3948	0.0657	3.1053	1.3405	1.1987	0.3101
0.30	0.008369	0.3688	0.0525	3.4884	1.3348	1.5040	0.3977
0.35	0.007785	0.3442	0.0516	3.8066	1.3855	1.8819	0.5323
0.40	0.007608	0.3740	0.0504	4.9534	1.6674	2.5261	0.7546
0.45	0.007070	0.4156	0.0519	6.8042	2.4790	4.0898	1.3212
0.50	0.007706	0.5078	0.0620	100.0000	100.0000	99.2402	99.5285

TAB. G.19: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 24 h - Niveau de Bruit = 0 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.050310	0.9312	0.3499	1.8819	1.0186	0.5325	0.1381
0.10	0.022276	0.5753	0.1663	2.0066	1.3079	0.6948	0.1882
0.15	0.018444	0.5312	0.1283	2.6105	1.2365	0.8403	0.2193
0.20	0.013401	0.5260	0.1046	3.2326	1.5895	1.0637	0.2856
0.25	0.011593	0.4701	0.0582	3.3768	1.2771	1.2858	0.3511
0.30	0.008150	0.3649	0.0588	3.4884	1.2843	1.5897	0.4277
0.35	0.008216	0.3585	0.0628	4.2132	1.5048	2.0520	0.5767
0.40	0.009077	0.4156	0.0533	5.4171	2.2643	2.8456	0.8814
0.45	0.007701	0.4208	0.0533	7.0262	2.8029	4.5573	1.5426
0.50	0.012847	0.5039	0.0911	100.0000	100.0000	99.1039	99.4463

TAB. G.20: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 24 h - Niveau de Bruit = 1 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.068210	1.1715	0.5531	2.1975	1.3765	0.5870	0.1784
0.10	0.034333	0.8273	0.2064	2.5507	1.2097	0.7429	0.2159
0.15	0.019963	0.6325	0.1185	2.7845	1.4166	0.9091	0.2729
0.20	0.014985	0.4247	0.1309	2.6300	1.6394	1.1390	0.3361
0.25	0.012524	0.4753	0.0928	3.2534	1.4872	1.3728	0.3940
0.30	0.009786	0.3857	0.0712	3.7937	1.5371	1.7040	0.4960
0.35	0.009200	0.3974	0.0643	4.2469	1.7014	2.1663	0.6586
0.40	0.007837	0.3935	0.0585	5.5002	2.2046	3.0339	1.0068
0.45	0.008135	0.4208	0.0559	7.8912	3.3212	5.1690	1.9014
0.50	0.017822	0.5520	0.1084	100.0000	100.0000	99.1039	99.4463

TAB. G.21: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 24 h - Niveau de Bruit = 2 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.054455	0.9039	0.5107	1.7728	1.2664	0.6416	0.2225
0.10	0.038851	0.7312	0.5200	2.2365	1.7679	0.8143	0.3274
0.15	0.016048	0.5455	0.1326	2.6001	1.0367	1.0130	0.2900
0.20	0.014341	0.4766	0.0983	2.9741	1.3953	1.2897	0.3859
0.25	0.013522	0.5013	0.1280	3.7209	1.9247	1.6079	0.5496
0.30	0.012344	0.4494	0.0856	4.1859	1.8267	1.9338	0.6378
0.35	0.012068	0.4390	0.1268	4.8872	2.5254	2.5066	0.9886
0.40	0.009993	0.4701	0.0692	6.7081	2.3888	3.7066	1.2241
0.45	0.015878	0.4364	0.1115	10.2913	9.9909	6.8327	5.3998
0.50	0.029736	0.9247	0.1631	100.0000	100.0000	99.1039	99.4463

TAB. G.22: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 24 h - Niveau de Bruit = 4 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.078690	1.0987	0.8451	1.9235	1.8267	0.7559	0.4358
0.10	0.055723	0.8624	0.5851	2.3598	1.9562	1.0104	0.5099
0.15	0.050927	0.8273	0.7007	3.1962	3.1667	1.3650	0.8232
0.20	0.027622	0.6065	0.2649	3.3703	2.4260	1.6143	0.7817
0.25	0.023317	0.7948	0.2796	5.2171	2.5804	2.3455	0.9370
0.30	0.019733	0.6338	0.1265	5.1638	3.0523	2.6702	1.2071
0.35	0.020070	0.6013	0.1473	6.1899	5.1171	3.5768	2.2130
0.40	0.029392	0.6935	0.1796	10.3433	12.5396	6.1886	6.3985
0.45	0.048713	0.8806	0.4093	25.3672	46.8180	17.3929	31.6193
0.50	0.173000	2.4689	0.6410	100.0000	100.0000	99.1039	99.4463

TAB. G.23: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 24 h - Niveau de Bruit = 8 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.176087	1.4819	1.7648	2.4624	2.8407	1.1403	0.9843
0.10	0.065496	1.2559	0.4796	3.0066	1.4584	1.4403	0.5940
0.15	0.127641	1.3897	1.4587	4.0144	4.9199	2.0287	2.0467
0.20	0.125636	1.3650	1.2783	5.4236	7.8955	3.0001	3.5189
0.25	0.159618	1.5585	1.5068	6.9210	11.9494	3.9820	5.6918
0.30	0.128126	1.2273	1.4918	9.3302	17.1552	5.5639	9.1375
0.35	0.280995	1.9546	2.1314	15.9669	33.3711	10.2056	20.2501
0.40	0.446180	3.1014	2.4346	32.0948	48.5350	21.9814	34.5101
0.45	0.816171	5.0054	2.9254	58.8919	74.4945	45.9628	63.4553
0.50	1.866330	8.1834	4.4928	100.0000	100.0000	99.1039	99.4463

TAB. G.24: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 24 h - Niveau de Bruit = 16 %

Marge	1-AUC	Erreur (%)		Query (%)		Apprentissage (%)	
(%)	(%)	Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.096709	1.6027	0.6237	3.0988	1.8060	0.4572	0.1078
0.10	0.039640	0.9052	0.3340	3.0573	1.9809	0.6273	0.1487
0.15	0.026516	0.8013	0.1568	3.4910	1.6665	0.7650	0.1871
0.20	0.016763	0.5247	0.1196	3.4923	2.0830	0.9637	0.2372
0.25	0.011258	0.4065	0.0738	3.4456	1.4521	1.1987	0.3026
0.30	0.011001	0.4364	0.0625	3.9495	1.6342	1.4559	0.3908
0.35	0.009455	0.3675	0.0562	3.9911	1.5898	1.8624	0.5292
0.40	0.008285	0.4013	0.0539	4.9612	1.8550	2.5053	0.7621
0.45	0.007925	0.4039	0.0539	6.6457	2.5516	4.0326	1.3252
0.48	0.007721	0.4286	0.0576	10.1030	4.0659	6.9665	2.5793
0.50	0.007934	0.5026	0.0625	100.0000	100.0000	99.0623	99.4011

Tab. G.25: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 48 h - Niveau de Bruit = 0 %

Marge	1-AUC	Erreur (%)		Query (%)		Apprentissage (%)	
(%)	(%)	Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.009113	0.4013	0.1386	0.8065	0.3698	0.2208	0.0496
0.10	0.050565	1.1026	0.5237	3.5105	2.0974	0.6987	0.1983
0.15	0.031000	0.8377	0.1698	3.5391	1.8740	0.8624	0.2332
0.20	0.013927	0.5312	0.1317	3.4144	1.8259	1.0247	0.2833
0.25	0.012195	0.4740	0.0723	3.6703	1.4719	1.2676	0.3450
0.30	0.011024	0.3753	0.0700	3.9885	1.6065	1.5871	0.4375
0.35	0.009660	0.4052	0.0553	4.4495	1.7875	2.0585	0.6070
0.40	0.009570	0.4468	0.0620	6.1301	2.2406	2.8287	0.8791
0.45	0.008185	0.4052	0.0591	7.0808	3.0148	4.3599	1.5800
0.48	0.008574	0.4325	0.0605	11.5940	5.9659	8.2042	3.6371
0.50	0.012019	0.5364	0.0850	100.0000	100.0000	99.0935	99.4247

Tab. G.26: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 48 h - Niveau de Bruit = 1 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.004610	0.2831	0.1035	0.6403	0.2827	0.2416	0.0669
0.10	0.041651	0.7948	0.5462	2.7014	2.0236	0.7143	0.2355
0.15	0.027738	0.6273	0.2072	2.9209	2.0236	0.8909	0.3032
0.20	0.016499	0.6143	0.1254	3.7988	1.6564	1.1598	0.3266
0.25	0.010467	0.4520	0.0876	3.7807	1.4820	1.4156	0.3903
0.30	0.011557	0.4299	0.0787	4.1833	2.1043	1.7390	0.5421
0.35	0.010925	0.4104	0.0663	4.5521	1.8708	2.1390	0.6877
0.40	0.009173	0.4117	0.0654	5.3612	2.4775	2.9495	1.0399
0.45	0.009588	0.4390	0.0588	8.5835	3.7411	5.3041	1.9830
0.48	0.010746	0.4481	0.0847	14.0148	12.1468	10.1108	7.0845
0.50	0.018362	0.6104	0.1110	100.0000	100.0000	99.0935	99.4247

TAB. G.27: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 48 h - Niveau de Bruit = 2 %

Marge (%)	1-AUC (%)	Erreur (%)		Query (%)		Apprentissage (%)	
		Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.162765	1.7949	1.2973	3.1521	2.5848	0.7065	0.3372
0.10	0.043022	0.7143	0.4594	2.3429	2.1931	0.8026	0.3580
0.15	0.030715	0.8286	0.2989	3.8456	1.8423	1.1702	0.3828
0.20	0.022700	0.5883	0.1819	3.5651	2.4360	1.3143	0.5176
0.25	0.016690	0.4870	0.1291	3.8274	2.3507	1.5299	0.5940
0.30	0.011558	0.4961	0.0865	4.1794	1.9190	1.9235	0.6168
0.35	0.011407	0.4455	0.0842	5.3872	2.4637	2.6235	0.9171
0.40	0.009966	0.4364	0.0804	7.3341	3.3096	3.9768	1.4731
0.45	0.021700	0.4883	0.1110	12.9836	15.2498	8.0899	7.6491
0.48	0.026904	0.6104	0.1986	30.3544	53.6774	22.1944	38.5931
0.50	0.034815	0.9728	0.1902	100.0000	100.0000	99.0935	99.4247

TAB. G.28: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 48 h - Niveau de Bruit = 4 %

Marge	1-AUC	Erreur (%)		Query (%)		Apprentissage (%)	
(%)	(%)	Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.173539	1.4715	1.8083	2.5676	3.3849	0.8689	0.6750
0.10	0.085007	1.2949	0.4574	3.1313	1.8446	1.0871	0.4978
0.15	0.044352	0.8416	0.3868	3.4988	2.2608	1.4429	0.6652
0.20	0.037274	0.7961	0.3998	4.0430	3.8665	1.7403	1.1189
0.25	0.036619	0.6507	0.5603	4.8794	5.3010	2.1455	1.6063
0.30	0.026995	0.5831	0.2891	5.3015	5.7837	2.6988	2.0905
0.35	0.026360	0.6611	0.2306	7.8003	7.6511	4.0625	3.1468
0.40	0.040746	0.7598	0.3816	12.2057	20.4562	7.0301	10.1912
0.45	0.073925	1.0974	0.5228	29.0310	50.5084	19.1930	34.0320
0.48	0.107431	1.5299	0.7021	60.7712	82.2173	48.3305	71.3272
0.50	0.164307	2.2897	0.8113	100.0000	100.0000	99.0935	99.4247

Tab. G.29: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 48 h - Niveau de Bruit = 8 %

Marge	1-AUC	Erreur (%)		Query (%)		Apprentissage (%)	
(%)	(%)	Ham	Spam	Ham	Spam	Ham	Spam
0.05	0.266080	2.2949	1.5734	3.7469	3.0341	1.6338	1.0555
0.10	0.244742	1.7312	1.9510	4.1054	5.5575	1.9377	2.0668
0.15	0.264354	1.4533	3.1284	4.6456	11.4928	2.3676	4.6504
0.20	0.285383	2.0884	2.1127	7.6756	12.4036	3.7924	5.4923
0.25	0.203216	1.5728	1.9789	7.9990	16.6185	4.2690	7.8739
0.30	0.260021	1.7988	2.4323	11.9836	25.2263	6.9016	13.3242
0.35	0.414701	2.5469	2.5784	19.6202	36.3942	11.7758	22.1371
0.40	0.542733	3.1274	2.9955	33.6714	53.9074	22.5437	38.2985
0.45	0.929184	4.9638	3.6180	58.0023	78.1353	45.4070	66.3047
0.48	1.370840	6.4587	4.1253	82.5617	94.0834	73.0483	89.5206
0.50	1.742530	7.5860	4.3977	100.0000	100.0000	99.0935	99.4247

Tab. G.30: Impact de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 48 h - Niveau de Bruit = 16 %

Marge	Taux d'Erreur					
	0.00	0.01	0.02	0.04	0.08	0.16
0.05	201694	202654	202469	204434	209423	215180
0.10	234168	233802	238545	237439	247846	254569
0.15	265583	266742	270213	273646	279157	303669
0.20	294582	294303	298586	306610	320316	341882
0.25	329719	335399	339669	344745	364793	393179
0.30	376430	378941	385755	394609	430330	458308
0.35	430121	438613	445185	454593	488246	573893
0.40	520542	524277	543609	564711	639609	853619
0.45	690996	717944	756977	815483	1009323	2312144
0.50	1881308	5043556	5616107	5897515	6006177	6056271

TAB. G.31: Dimension du classificateur en fonction du niveau de bruit et de la marge d'apprentissage actif - Retour d'information d'étiquette immédiat

Marge	Taux d'Erreur					
	0.00	0.01	0.02	0.04	0.08	0.16
0.05	188812	194741	198964	209225	227982	256417
0.10	223078	225786	230495	245918	261054	314054
0.15	255691	260323	263747	280658	306466	371998
0.20	286524	290584	295418	313742	343849	469192
0.25	320871	325549	333934	358605	382649	621422
0.30	365100	370731	382723	403856	483023	745267
0.35	417362	423415	444134	464402	592077	1138286
0.40	504148	522227	537598	582455	818189	2136927
0.45	667548	694548	759601	944303	1929030	3725376
0.50	1878888	5183807	5611743	5881897	6003479	6045369

TAB. G.32: Dimension du classificateur en fonction du niveau de bruit et de la marge d'apprentissage actif -Retour d'information aléatoire de distribution exponentielle de moyenne 6 h

Marge	Taux d'Erreur					
	0.00	0.01	0.02	0.04	0.08	0.16
0.05	172012	185336	207975	224198	287792	426035
0.10	206730	221784	239593	272504	340675	436914
0.15	232647	248945	276794	285806	402409	644857
0.20	267474	286144	299502	327822	448145	731275
0.25	301880	320581	339021	394193	515162	1001495
0.30	350508	364630	390137	427736	590841	1179922
0.35	403736	425552	447267	529277	724706	1787143
0.40	482257	519990	558537	661799	1103032	2501197
0.45	643995	701807	796300	1147908	2285049	3891660
0.50	1874602	5162159	5587746	5856444	5976767	6018613

Tab. G.33: Dimension du classificateur en fonction du niveau de bruit et de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 24 h

Marge	Taux d'Erreur					
	0.00	0.01	0.02	0.04	0.08	0.16
0.05	160730	225777	230418	253756	315552	453254
0.10	200313	223056	233796	286107	351431	555232
0.15	229449	249666	270890	314559	421215	770246
0.20	263561	285436	302638	352613	468483	873138
0.25	302999	320919	332159	387327	539511	1171966
0.30	344817	366903	393470	441508	638610	1425569
0.35	399405	424833	462474	516149	807000	1995839
0.40	482537	518578	571470	677395	1285783	2730083
0.45	644454	702513	805565	1174712	2599858	4080570
0.50	1871860	5154773	5580085	5848814	5969057	6010944

Tab. G.34: Dimension du classificateur en fonction du niveau de bruit et de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 48 h

Marge	Taux d'Erreur					
	0.00	0.01	0.02	0.04	0.08	0.16
0.05	0.011662	0.012297	0.010751	0.013613	0.015706	0.016013
0.10	0.008518	0.009229	0.007810	0.010157	0.011701	0.013290
0.15	0.007193	0.007448	0.008073	0.008014	0.009363	0.011702
0.20	0.006450	0.007581	0.006616	0.008063	0.008416	0.011803
0.25	0.006129	0.006200	0.006383	0.007366	0.007889	0.009316
0.30	0.005861	0.006155	0.005940	0.005570	0.007209	0.008938
0.35	0.005481	0.005523	0.005621	0.006335	0.007076	0.009831
0.40	0.005113	0.005190	0.005972	0.006015	0.007056	0.013703
0.45	0.005025	0.005067	0.005509	0.006259	0.008575	0.098798
0.50	0.005588	0.010877	0.016021	0.037899	0.218627	2.050730

TAB. G.35: Erreur de classement (1-AUC%) en fonction du niveau de bruit et de la marge d'apprentissage actif - Retour d'information d'étiquette immédiat

Marge	Taux d'Erreur					
	0.00	0.01	0.02	0.04	0.08	0.16
0.05	0.014408	0.014517	0.016900	0.019054	0.024396	0.030288
0.10	0.011173	0.014873	0.015027	0.012336	0.015106	0.048310
0.15	0.009054	0.011078	0.009888	0.012366	0.015494	0.024441
0.20	0.009085	0.008500	0.008422	0.009502	0.012361	0.020141
0.25	0.008709	0.008160	0.007612	0.009431	0.011394	0.040773
0.30	0.006901	0.007720	0.008030	0.008658	0.012899	0.029299
0.35	0.006785	0.006400	0.006762	0.007770	0.011843	0.055315
0.40	0.006188	0.006666	0.006658	0.007688	0.013718	0.206590
0.45	0.006100	0.006479	0.006416	0.008161	0.040979	0.626636
0.50	0.006620	0.011918	0.017913	0.034010	0.164003	1.885750

TAB. G.36: Erreur de classement (1-AUC%) en fonction du niveau de bruit et de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 6 h

Marge	Taux d'Erreur					
	0.00	0.01	0.02	0.04	0.08	0.16
0.05	0.053748	0.050310	0.068210	0.054455	0.078690	0.176087
0.10	0.026610	0.022276	0.034333	0.038851	0.055723	0.065496
0.15	0.021507	0.018444	0.019963	0.016048	0.050927	0.127641
0.20	0.014277	0.013401	0.014985	0.014341	0.027622	0.125636
0.25	0.010668	0.011593	0.012524	0.013522	0.023317	0.159618
0.30	0.008369	0.008150	0.009786	0.012344	0.019733	0.128126
0.35	0.007785	0.008216	0.009200	0.012068	0.020070	0.280995
0.40	0.007608	0.009077	0.007837	0.009993	0.029392	0.446180
0.45	0.007070	0.007701	0.008135	0.015878	0.048713	0.816171
0.50	0.007706	0.012847	0.017822	0.029736	0.173000	1.866330

Tab. G.37: Erreur de classement (1-AUC%) en fonction du niveau de bruit et de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 24 h

Marge	Taux d'Erreur					
	0.00	0.01	0.02	0.04	0.08	0.16
0.05	0.096709	0.104602	0.068518	0.162765	0.173539	0.266080
0.10	0.039640	0.050565	0.041651	0.043022	0.085007	0.244742
0.15	0.026516	0.031000	0.027738	0.030715	0.044352	0.264354
0.20	0.016763	0.013927	0.016499	0.022700	0.037274	0.285383
0.25	0.011258	0.012195	0.010467	0.016690	0.036619	0.203216
0.30	0.011001	0.011024	0.011557	0.011558	0.026995	0.260021
0.35	0.009455	0.009660	0.010925	0.011407	0.026360	0.414701
0.40	0.008285	0.009570	0.009173	0.009966	0.040746	0.542733
0.45	0.007925	0.008185	0.009588	0.021700	0.073925	0.929184
0.50	0.007934	0.012019	0.018362	0.034815	0.164307	1.742530

Tab. G.38: Erreur de classement (1-AUC%) en fonction du niveau de bruit et de la marge d'apprentissage actif - Retour d'information aléatoire de distribution exponentielle de moyenne 48 h

ANNEXE H

Publications

Revues

J. M. Martins da Cruz, *Mail filtering on medium/huge mail servers*, Computational Methods in Science and Technology 11 (2005), no. 2, 101-108.

Conférences

L. Aublet-Cuvelier and J. M. Martins da Cruz, *Les défis et les opportunités techniques du fonctionnement d'un service antispam mutualisé* - JRES 2011 - Toulouse, 2011.

J. M. Martins da Cruz, *Méthodologie d'évaluation de filtres anti-spam* - JRES 2009 - Nantes, 2009.

G. V. Cormack and J. M. Martins da Cruz, *On the relative age of spam and ham training samples for email filtering*, SIGIR '09 : Proceedings of the 32nd international ACM conference on Research and Development in Information Retrieval (New York, NY, USA), ACM, 2009.

J. M. Martins da Cruz and G. V. Cormack, *Using old spam and ham samples to train email filters*, Proc. CEAS 2009 - Sixth Conference on Email and Anti-Spam (Mountain View, CA), 2009.

J. M. Martins da Cruz, *Filtrage de mail sur des gros serveurs* - JRES 2005 - Marseille, 2009

J. M. Martins da Cruz, *sendmail X - la nouvelle génération de sendmail* - JRES 2005 - Marseille, 2005

J. M. Martins da Cruz, *Mail filtering on medium/huge mail servers with j-chkmail* - TERENA Networking Conference 2005 - Poznan, May 2005

Orateur invité

libmilter implementation : "pool of workers" or "one thread per connection"? - sendmail Meeting of The Minds - Jul 2003 - Oakland, CA, July 2003

Connection Rate Control with j-chkmail - sendmail Meeting of The Minds - Jul 2003 - Oakland, CA, July 2003

La lutte contre le spam dans une organisation importante - Seminaires Aristote, Ecole Polytechnique - Paris, Janvier 2005

Joe's j-chkmail, Solutions Linux 2005 - Paris, Janvier 2005

ANNEXE I

Distinctions et Prix

I.1 Sendmail - 25 Years of Trusted Messaging

FIG. I.1: Sendmail Award

http://www.sendmail.com/pdfs/pressreleases/Sendmail%20Innovation%20Awards_10%2025%2006_FINAL.pdf

MOUNTAIN VIEW, Calif. - October 25th, 2006 - Today at its 25 Years of Internet Mail celebration event, taking place at the Computer History Museum in Mountain View, California, Sendmail, Inc., the leading global provider of trusted messaging, announced the recipients of its inaugural Innovation Awards. Eight recipients from across the globe were recognized for the dramatic impact they have made to Internet communications and security using Sendmail technology. ...

The Sendmail Innovation Awards honored companies in three categories : Innovative Use of the Sendmail Mail Transfer Agent (MTA) ; Innovative Open Source Contribution and Sendmail Milter Innovation. Drawing from Sendmail's roots in innovation and with a vision for the future of secure and trusted messaging, Sendmail selected the award winners after evaluating hundreds of enterprise deployment scenarios and contributions to Open Source. Each of the winners demonstrated excellence

in developing one-of-a-kind Internet Mail solutions showing unusual creativity, providing best practice standards and providing measurable impact on solving real-world problems.

...

Innovative Open Source Contribution

EMEA Winner : Jose Marcio Martins da Cruz, Ecole des Mines de Paris - for his contributions to libmilter (the library that implements the Milter protocol and provides the API to applications). This contribution delivered an alternative implementation of the threading model, providing improved performance for operating systems such as Linux.

I.2 Terena Networking Conference 2005 - Selected Papers

`http://www.terena.org/publications/tnc2005-proceedings/`

The papers published here represent the best of the papers presented at the TERENA Networking Conference 2005 in Poznan, Poland. Over one hundred and sixty extended abstracts were submitted in response to the call for papers, and after review, sixty-two of these were chosen by the Programme Committee for presentation at the conference. In addition, seventy-two recognised experts in their field were invited to give presentations at the conference.

All speakers were then invited to submit full papers for potential selection for this publication of the conference proceedings, and after review by the Programme Committee and other experts when necessary, those that were finally selected are published here.

The papers represent a good mix of subjects covering the entire breadth of the main themes of the conference.

The programme committee was composed of members of the TERENA community in Europe and from our sister organisation in the United States.

Olivier Martin
Programme Committee Chair

Les 10 papiers choisis ont constitué un numéro de la revue **Computational Methods in Science and Technology** :

J. M. Martins da Cruz, *Mail filtering on medium/huge mail servers*, Computational Methods in Science and Technology 11 (2005), no. 2, 101-108.

ANNEXE J

Le logiciel de filtrage de spam *j-chkmail*

j-chkmail est un logiciel de filtrage de spams, destiné à des serveurs de messagerie de taille moyenne ou importante. C'est le résultat pratique des travaux de l'auteur dans le domaine de la messagerie électronique depuis 2001.

Il s'agit d'un logiciel distribué sous licence libre du type GPL, écrit en langage C avec, comme principe, d'implémenter un nombre minimal de méthodes de filtrage sélectionnées entre celles qui sont, à la fois, efficaces et peu consommatrices de ressources informatiques.

Il regroupe deux familles de méthodes de filtrage :

Filtrage protocolaire - Ce sont des heuristiques basées uniquement dans les caractéristiques de la connexion et dépendantes du client SMTP qui se connecte sur le serveur de messagerie. Ce sont des heuristiques telles :

- **Limitations à court terme :** évaluées dans une fenêtre temporelle de taille 10 minutes. Ce sont des limitations du type cadence de connexion, nombre de messages ou destinataires. Cette catégorie de limitation sert, d'une part à protéger le serveur contre des attaques de déni de service et, d'autre part à éviter les flots de messages inattendus envoyés par des outils "non-humains" ;
- **Limitations à long terme :** évaluées dans une fenêtre temporelle de taille 4 heures. Ce sont des limitations visant à détecter les clients SMTP avec un comportement douteux tel des erreurs répétitifs dans les adresses destinataires, ou alors le moissonnage d'adresses par balayage exhaustif d'un dictionnaire de noms ;
- **Greylisting** : c'est une méthode permettant de détecter des *botnets* - des ordinateurs personnels infectés par un logiciel malveillant et utilisés pour distribuer des spams.

Outre la protection du serveur contre les attaques de déni de servie, le filtrage protocolaire vise surtout à éliminer les messages dont le filtrage est trivial et diminuer la charge de traitement qui serait imposée au filtrage de contenu.

Filtrage de contenu - Le filtrage de contenu examine le message entier (corps et entêtes) à l'aide de deux classificateurs statistiques : un Bayésien Naif et le classificateur discriminant linéaire. Ce dernier a été développé pour les besoins de cette thèse.

Les scores des deux classificateurs sont combinés de façon triviale en un score final : moyenne des scores en échelle *logit*.

L'apprentissage des deux classificateurs se fait hors ligne sur une fenêtre temporelle glissante : apprentissage supervisé pour le classificateur Bayésien Naïf et apprentissage actif pour le classificateur discriminant linéaire. Les classificateurs résultant de cet apprentissage sont distribués à des

utilisateurs externes (des organismes d'enseignement et recherche français), pour leur épargner la tâche de collecte d'exemples d'apprentissage.

Il n'y a pas d'évaluation d'efficacité objective possible dans ce contexte d'utilisation mais, vus les retours d'information et l'efficacité ressentie par les utilisateurs, il semblerait que la qualité de filtrage soit comparable à celle des meilleurs filtres du marché.

Bien entendu, l'intégration de résultats de cette thèse dans ce filtre, est prévue dans un proche avenir.

ANNEXE K

Glossaire

Statistique Textuelle

Caractère - signe typographique utilisé pour l'encodage du texte sur un support lisible par l'ordinateur.

Caractères délimiteurs/non-délimiteurs - distinction opérée sur l'ensemble des caractères, qui entrent dans la composition du texte permettant aux procédures informatisées de segmenter le texte en occurrences (suite de caractères non délimiteurs bornée à ses extrémités par des caractères délimiteurs). On distingue les délimiteurs de forme (le blanc, les signes de ponctuation usuels, et les éventuels signes de pré-analyse), les délimiteurs de séquence (ponctuations faibles et fortes contenues dans la police de caractères) et les délimiteurs de phrase (sous-ensemble des délimiteurs de séquence - ponctuations fortes).

Corpus - ensemble de textes réunis à des fins de comparaison ; servant de base à une étude quantitative.

Hapax - forme dont la fréquence est égale à un dans le corpus (hapax du corpus) ou dans une de ses parties (hapax de la partie).

Forme - ou *forme graphique* archétype correspondant aux occurrences identiques dans un corpus de textes, c'est à dire aux occurrences composées strictement des mêmes caractères non-délimiteurs d'occurrence

Forme banale - (sp) pour une partir du corpus donnée, forme ne présentant aucune spécificité (ni positive ni négative) dans cette partie.

Lemmatisation - regroupement sous une forme canonique (en général à partir d'un dictionnaire) des occurrences du texte. En français, ce regroupement se pratique en général de la manière suivante : les formes verbales à l'infinitif, les substantifs au singulier, les adjectifs au masculin singulier et les formes élidées à la forme sans élision.

Lexique - ensemble virtuel des mots d'une langue

Lexicométrie - ensemble de méthodes permettant d'opérer des réorganisations formelles de la séquence textuelle et des analyses statistiques portant sur le vocabulaire d'un corpus de textes.

Occurrence - suite de caractères non délimiteurs bornée à ses extrémités par deux caractères délimiteurs de forme.

Polyforme - archétype des occurrences d'un segment ; suite de formes non séparées par un séparateur de séquence, qui n'est pas obligatoirement attestée sans le corpus.

Segment - toute suite d'occurrences consécutives dans le corpus et non séparées par un séparateur de séquence est un segment du texte.

Segmentation - opération qui consiste à délimiter des unités minimales dans un texte

Segmentation automatique - ensemble d'opérations réalisées au moyen de procédures informatisées qui aboutissent à découper, selon des règles prédéfinies, un texte stocké sur un support lisible par un ordinateur en unités distinctes que l'on appelle des unités minimales.

Séquence - suite d'occurrences du texte non séparées par un délimiteur de séquence.

Terme - nom générique s'appliquant à la fois aux formes et aux polyformes. Dans le premier cas on parlera de termes de longueur 1. Les polyformes sont des termes de longueur 2, 3, etc.

Unités minimales - (pour un type de segmentation) unités que l'on ne décompose pas en unités plus petites pouvant entrer dans leur composition.

Vocabulaire - ensemble des formes attestées dans un corpus de textes.

Vocabulaire de base - ensemble des formes du corpus ne présentant, pour un seuil fixé, aucune spécificité (négative ou positive) dans aucune des parties, (i.e., l'ensemble des formes banales).

Autres

DARPA - Defense Advanced Research Projects Agency

Filoutage - (**Phishing**) Message piège invitant le destinataire à dévoiler des donné

Faux Négatif - Um spam classé comme étant du ham

Faux Positif - Un ham classé comme étant du spam

FSF - Free Software Foundation - Entité à l'origine de la licence GPL. Voir : `http://www.fsf.org`.

GPL - General Public License - Licence d'utilisation typique des logiciels libres (ce n'est pas la seule), promue par la FSF.

Ham - Un message légitime

HMR - Ham Misclassification Rate - False Positive Rate - Taux de Faux Positifs

HTML - Hypertext Mark Up Language -

ISP - Internet Service Provider

LAM - Logistic Average Misclassification

MIME - Multipurpose Internet Mail Extensions

MDA - Mail Delivery Agent - logiciel utilisé pour remettre les messages dans la boîte aux lettres des utilisateurs. Lorsque le MTA reçoit un message destiné à un utilisateur local, ce message est transmis au MDA.

MTA - Mail Transport Agent - logiciel de routage de messages - souvent appelé *Serveur de Messagerie*

MUA - Mail User Agent

NLP - Natural Language Processing - Traitement d'objets textuels

RBL - Real Time Blackhole List

ROC - Receiver Operating Caracteristic

1-AUC -

SMR - Spam Misclassification Rate - False Negative Rate - Taux de Faux Negatif -

SMTP - Simple Mail Transport Protocol

Spam - Un message non souhaité. Voir définition dans le Chapitre 1.

TFN / TFN - Taux de Faux Positifs / Taux de Faux Négatifs

Bibliographie

[1] Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch, *Replacing suffix trees with enhanced suffix arrays*, Journal of Discrete Algorithms **2** (2004), no. 1, 53 – 86, The 9th International Symposium on String Processing and Information Retrieval.

[2] Charu C. Aggarwal, *On change diagnosis in evolving data streams*, IEEE Trans. on Knowl. and Data Eng. **17** (2005), no. 5, 587–600.

[3] Charu C. Aggarwal, T. J. Watson, Resch Ctr, Jiawei Han, Jianyong Wang, and Philip S. Yu, *A framework for clustering evolving data streams*, In VLDB, 2003, pp. 81–92.

[4] S. M. Ali and S. D. Silvey, *A general class of coefficients of divergence of one distribution from another*, Journal of Royal Statistics Society **28** (1966), 131–142.

[5] Ethem Alpaydin, *Introduction to machine learning*, MIT Press, Cambridge, MA, USA, 2004.

[6] B.M. Amine and M. Mimoun, *Wordnet based cross-language text categorization*, Computer Systems and Applications, 2007. AICCSA '07. IEEE/ACS International Conference on, May 2007, pp. 848–855.

[7] I. Androutsopoulos, J. Koutsias, K. Chandrinos, and C. D. Spyropoulos, *An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages*, SIGIR '00 : Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 2000, pp. 160–167.

[8] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, K. V. Ch, G. Paliouras, and C. D. Spyropoulos, *An evaluation of naive bayesian anti-spam filtering*, In Proceedrings of the Workshop on Machine Learning in the New Information Age - 11th European Conference on Machine Learning (ECML 2000, 2000, pp. 9–17.

[9] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. D. Spyropoulos, and P. Stamatopoulos, *Learning to filter spam e-mail : A comparison of a naive bayesian and a memory-based approach*, Proceedings of the Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000), 2000, pp. 1–13.

[10] I. Androutsopoulos, G. Paliouras, and E. Michelakis, *Learning to filter unsolicited commercial E-Mail*, Tech. Report 2004/2, NCSR "Demokritos", October 2004.

[11] Geoff Hulten Anthony, Anthony Penta, Gopalakrishnan Seshadrinathan, and Manav Mishra, *Trends in spam products and methods*, CEAS 2004 : Proceedings of the 1st Conference on Email and Anti-Spam, 2004.

[12] Hrishikesh B. Aradhye, Gregory K. Myers, and James A. Herson, *Image analysis for efficient categorization of image-based spam e-mail*, ICDAR '05 : Proceedings of the Eighth International Conference on Document Analysis and Recognition (Washington, DC, USA), IEEE Computer Society, 2005, pp. 914–918.

[13] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar, *Can machine learning be secure ?*, ASIACCS '06 : Proceedings of the 2006 ACM Symposium on Information, computer and communications security (New York, NY, USA), ACM, 2006, pp. 16–25.

[14] M. Basseville, *Distance measures for signal processing and pattern recognition*, Signal Process. **18** (1989), no. 4, 349–369.

[15] R. Battiti, *Using mutual information for selecting features in supervised neural net learning*, Neural Networks, IEEE Transactions on **5** (1994), no. 4, 537–550.

[16] Jan Beirlant, Luc Devroye, László Györfi, and Igor Vajda, *Large deviations of divergence measures on partitions*, Journal of Statistical Planning and Inference **93** (2001), no. 1-2, 1 – 16.

[17] Ron Bekkerman, *Distributional clustering of words for text categorization*, Master's thesis, Technion - Israel Institute of Technology, Haifa, 2003.

[18] A. Benveniste, M. Métivier, and P. Priouret, *Algorithmes adaptatifs et approximations stochastiques*, Masson, Paris, FR, 1987.

[19] Helmut Berger, Michael Dittenbach, and Dieter Merkl, *Analyzing the effect of document representation on machine learning approaches in multi-class e-mail filtering*, WI '06 : Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (Washington, DC, USA), IEEE Computer Society, 2006, pp. 297–300.

[20] Steffen Bickel, Michael Brückner, and Tobias Scheffer, *Discriminative learning under covariate shift*, J. Mach. Learn. Res. **10** (2009), 2137–2155.

[21] Battista Biggio, Giogio Fumera, Ignazio Pillai, and Fabio Roli, *Image spam filtering by content obscuring detection*, CEAS 2007 – The Third Conference on Email and Anti-Spam, 2007.

[22] G. Birkhoff and S. Mac Lane, *Algebra*, Gauthier-Villars, 1971.

[23] Christopher M. Bishop, *Pattern recognition and machine learning*, Springer Science, New York, NY, 2006.

[24] Ismaïl Biskri and Sylvain Delisle, *Text classification and multilinguism : Getting at words via n-grams of characters*, 2002.

[25] Avrim L. Blum and Pat Langley, *Selection of Relevant Features and Examples in Machine Learning*, Artificial Intelligence **97** (1997), no. 1-2, 245 – 271, Relevance.

[26] Thomas Scott Blyth, *Lattices and ordered algebraic structures*, 1st ed., Springer-Verlag, London, 2005.

[27] Mirko Boettcher, Frank Hoeppner, and Myra Spiliopoulou, *On exploiting the power of time in data mining*, SIGKDD Explorations Newsletter **10** (2008), no. 2, 3–11.

[28] Geert Booij, *The grammar of words - an introduction to linguistic morphology*, Oxford Textbooks in Linguistics, Oxford University Press, 2007.

[29] Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou, *Fast kernel classifiers with online and active learning*, J. Mach. Learn. Res. **6** (2005), 1579–1619.

[30] Léon Bottou, *Une approche théorique de l'apprentissage connexionniste : Applications à la reconnaissance de la parole*, Ph.D. thesis, Université de Paris XI, Orsay, France, 1991.

[31] Léon Bottou, *Online algorithms and stochastic approximations*, Online Learning and Neural Networks (David Saad, ed.), Cambridge University Press, Cambridge, UK, 1998.

[32] Léon Bottou and Yann LeCun, *Large scale online learning*, Advances in Neural Information Processing Systems 16 (Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, eds.), MIT Press, Cambridge, MA, 2004.

[33] Léon Bottou and Olivier Bousquet, *The tradeoffs of large scale learning*, In : Advances in Neural Information Processing Systems 20, 2008, pp. 161–168.

[34] Stéphane Boucheron, Olivier Bousquet, and Gábor Lugosi, *Theory of classification : a survey of some recent advances*, ESAIM : P&S **9** (2005), 323–375.

[35] A. Bratko, G. V. Cormack, B. Filipič, T. R. Lynam, and B. Zupan, *Spam filtering using statistical data compression models*, Journal of Machine Learning Research **7** (2006), no. Dec, 2673–2698.

[36] A. Bratko, B. Filipic, and B. Zupan, *Towards Practical PPM Spam Filtering : Experiments for the TREC 2006 Spam Track*, Proc. 15th Text REtrieval Conference (TREC 2006) (Gaithersburg, MD), November 2006.

[37] Leo Breiman, *Statistical modeling : The two cultures*, Statistical Science **16** (2001), no. 3, 199–231.

[38] Peter J. Brockwell and Richard A. Davis, *Introduction to Time Series and Forecasting*, Springer, March 2002.

[39] Peter F. Brown, Vincent J. Della Pietra, Robert L. Mercer, Stephen A. Della Pietra, and Jennifer C. Lai, *An estimate of an upper bound for the entropy of english*, Comput. Linguist. **18** (1992), no. 1, 31–40.

[40] Stefan Büttcher, Charles L. A. Clarke, and Gordon V. Cormack, *Information retrieval - implementing and evaluating search engines*, The MIT Press, Cambridge, Massachussets, 2010.

[41] Byungki Byun, Chin-Hui Lee, Steve Webb, and Calton Pu, *A discriminative classifier learning approach to image modeling and spam image identification*, CEAS 2007 – The Third Conference on Email and Anti-Spam, 2007.

[42] Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani, *A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization*, Text databases & document management : theory & practice (2001), 78–102.

[43] X. Carreras and L. Márquez, *Boosting trees for anti-spam email filtering*, Proc. of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing, 2001.

[44] William Cavnar, , William B. Cavnar, and John M. Trenkle, *N-gram-based text categorization*, Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, 1994, pp. 161–175.

[45] William B. Cavnar and Alan J. Vayda, *N-gram-based matching for multi-field database access in postal applications*, Proceedings of the 1993 Symposium on Document Analysis and Information Retrieval (Las Vegas), University of Nevada, 1993.

[46] Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile, *On the generalization ability of on-line learning algorithms*, IEEE Transactions on Information Theory **50** (2004), no. 9, 2050–2057.

[47] Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni, *Worst-case analysis of selective sampling for linear classification*, J. Mach. Learn. Res. **7** (2006), 1205–1230.

[48] Chris Chatfield, *An introduction to time series analysis*, 6th ed., Chapman & Hall CRC, Boca Raton, FL, US, 2003.

[49] Jean-Paul Chiles and Pierre Delfiner, *Geostatistics : Modeling spatial uncertainty*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons Inc, New York, NY, USA, 1999.

[50] Abdur Chowdhury, Ophir Frieder, David Grossman, and Mary Catherine McCabe, *Collection statistics for fast duplicate document detection*, ACM Trans. Inf. Syst. **20** (2002), 171–191.

[51] Kenneth W. Church and William A. Gale, *Enhanced good-turing and cat-cal : two new methods for estimating probabilities of english bigrams*, HLT '89 : Proceedings of the workshop on Speech and Natural Language (Morristown, NJ, USA), Association for Computational Linguistics, 1989, pp. 82–91.

[52] Kenneth Ward Church and Patrick Hanks, *Word association norms, mutual information, and lexicography*, Comput. Linguist. **16** (1990), no. 1, 22–29.

[53] Ali Ciltik and Tunga Gungor, *Time-efficient spam e-mail filtering using n-gram models*, Pattern Recognition Letters **29** (2008), no. 1, 19–33.

[54] W. W. Cohen, *Learning rules that classify e-mail*, In Papers from the AAAI Spring Symposium on Machine Learning in Information Access, AAAI Press, 1996, pp. 18–25.

[55] William W. Cohen, *Fast effective rule induction*, Proc. of the 12th International Conference on Machine Learning (Tahoe City, CA) (Armand Prieditis and Stuart Russell, eds.), Morgan Kaufmann, July 9–12, 1995, pp. 115–123.

[56] Gordon V. Cormack, *TREC 2006 Spam Track Overview*, Fifteenth Text REtrieval Conference (TREC-2006) (Gaithersburg, MD), NIST, 2006.

[57] ———, *TREC 2007 Spam Track Overview*, Sixteenth Text REtrieval Conference (TREC-2007) (Gaithersburg, MD), NIST, 2007.

[58] ———, *University of Waterloo Participation in the TREC 2007 Spam Track*, Sixteenth Text REtrieval Conference (TREC-2007) (Gaithersburg, MD), NIST, 2007.

[59] ———, *Email spam filtering : A systematic review*, vol. 1, Now Publishers, 2008.

[60] Gordon V. Cormack and Andrej Bratko, *Batch and on line filter comparison*, Proc. CEAS 2006 – Third Conference on Email and Anti-Spam (Mountain View, CA), 2006.

[61] ———, *Batch and on-line spam filter evaluation*, CEAS 2006 : The Third Conference on Email and Anti-Spam, 2006.

[62] Gordon V. Cormack and Aleksander Kolcz, *Spam filter evaluation with imprecise ground truth*, SIGIR '09 : Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 2009, pp. 604–611.

[63] Gordon V. Cormack and Thomas R. Lynam, *TREC spam filter evaluation toolkit*, http ://-plg.uwaterloo.ca/~gvcormac/jig/.

[64] ———, *Spam corpus creation for TREC*, CEAS 2005 : The Second Conference on E-mail and Anti-spam, 2005.

[65] ———, *TREC 2005 spam corpus*, http ://plg.uwaterloo.ca/~gvcormac/treccorpus, 2005.

[66] ———, *TREC 2005 Spam Track overview*, http ://-plg.uwaterloo.ca/~gvcormac/trecspamtrack05, 2005.

[67] ———, *Statistical precision of information retrieval evaluation*, SIGIR '06 : Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM Press, 2006, pp. 533–540.

[68] ———, *TREC 2006 spam corpora*, http ://plg.uwaterloo.ca/~gvcormac/treccorpus06, 2006.

[69] Gordon V. Cormack and Thomas R. Lynam, *Online supervised spam filter evaluation*, ACM Trans. Inf. Syst. **25** (2007), no. 3, 11.

[70] Gordon V. Cormack and Thomas R. Lynam, *TREC 2007 spam corpus*, http ://-plg.uwaterloo.ca/~gvcormac/treccorpus07, 2007.

[71] Gordon V. Cormack and J. M. Martins da Cruz, *On the relative age of spam and ham training samples for email filtering*, SIGIR '09 : Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 2009, pp. 744–745.

[72] T. Cover and R. King, *A convergent gambling estimate of the entropy of english*, IEEE Transactions on Information Theory **24** (1978), no. 4, 413–421.

[73] T. M. Cover and J. A. Thomas, *Elements of information theory*, 2nd ed., Wiley Interscience, New York, 2006.

[74] Paul S. P. Cowpertwait and Andrew V. Metcalfe, *Introductory time series with R*, Use R !, Springer, 2009.

[75] Michael J. Crawley, *Statistics - an introduction using r*, John Wiley and Sons, Ltd, West Sussex, England, 2010.

[76] Noel A. C. Cressie, *Statistical for spatial data*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons Inc, New York, NY, USA, 1993.

[77] Nello Cristianini and John Shawe-Taylor, *An Introduction to Support Vector Machines : and other kernel-based learning methods*, Cambridge University Press, New York, NY, USA, 2000.

[78] I. Csiszár, *Information-type measures of difference of probability distributions and indirect observations*, Studia Sci. Math. Ungar. **2** (1967), 299–318.

[79] ———, *Information theory and statistics : A tutorial*, vol. 1, Now Publishers, 2004.

[80] Padraig Cunningham, Niamh Nowlan, Sarah Jane Delany, and Mads Haahr, *A case-based approach to spam filtering that can track concept drift*, In The ICCBR'03 Workshop on Long-Lived CBR Systems, 2003, pp. 03–2003.

[81] Jose M. Martins da Cruz and Gordon V. Cormack, *Using old spam and ham samples to train email filters*, Proc. CEAS 2009 – Sixth Conference on Email and Anti-Spam (Mountain View, CA), 2009.

[82] N. Dalvi, P. Domingos, S. Mausam, S. Sanghai, and D. Verma, *Adversarial classification*, KDD '04 : Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (New York, NY, USA), ACM, 2004, pp. 99–108.

[83] Raymond J. D'Amore and Clinton P. Mah, *One-time complete indexing of text : theory and practice*, SIGIR '85 : Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 1985, pp. 155–164.

[84] J. N. Darroch and D. Ratcliff, *Generalized iterative scaling for log-linear models*, The Annals of Mathematical Statistics **43** (1972), no. 5, 1470–1480.

[85] Gerard de Melo and Stefan Siersdorfer, *Multilingual text classification using ontologies*, Proceedings of the 29th European Conference on Information Retrieval (ECIR 2007) (Rome, Italy) (Gianni Amati, Claudio Carpineto, and Giovanni Romano, eds.), Lecture Notes in Computer Science, vol. 4425, Springer, 2007, Acceptance Ratio 1 :4, pp. 541–548.

[86] S. J. Delany, P. Cunningham, and A. Tsymbal, *A comparison of ensemble and case-base maintenance techniques for handling concept drift in spam filtering*, Proceedings of the 19th International Conference on Artificial Intelligence (FLAIRS 2006) (G. Sutcliffe and R. Goebel, eds.), AAAI Press, 2006, pp. 340–345.

[87] Sarah Jane Delany, Padraig Cunningham, and Lorcan Coyle, *Case-based reasoning for spam filtering*, Artificial Intelligence Review **24** (2005), no. 3-4, 359–378.

[88] Luc Devroye, Laszlo Gyorfi, and Gabor Lugosi, *A probabilistic theory of pattern recognition*, Applications of Mathematics, vol. 31, Springer-Verlag, New York, NY, 1996.

[89] George Doddington, *Automatic evaluation of machine translation quality using n-gram co-occurrence statistics*, Proceedings of the second international conference on Human Language Technology Research (San Francisco, CA, USA), Morgan Kaufmann Publishers Inc., 2002, pp. 138–145.

[90] Pedro Domingos and Michael Pazzani, *On the optimality of the simple bayesian classifier under zero-one loss*, Mach. Learn. **29** (1997), no. 2-3, 103–130.

[91] Mark Dredze, Reuven Gevaryahu, and Ari Elias-Bachrach, *Learning fast classifiers for image spam*, CEAS 2007 – The Third Conference on Email and Anti-Spam, 2007.

[92] H. Drucker, Donghui Wu, and V. N. Vapnik, *Support vector machines for spam categorization*, Neural Networks, IEEE Transactions on **10** (1999), no. 5, 1048–1054.

[93] Marie Duflo, *Random iterative models*, 1st ed., Stochastic Modelling and Applied Probability, vol. 34, Springer-Verlag, New York, NY, 1997.

[94] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami, *Inductive learning algorithms and representations for text categorization*, CIKM '98 : Proceedings of the seventh international conference on Information and knowledge management (New York, NY, USA), ACM, 1998, pp. 148–155.

[95] J. Durbin and S. J. Koopman, *Time-series analysis by state space models*, vol. 24, Oxford Statistical Science Series, no. 24, Oxford University Press, 2008.

[96] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*, Chapman & Hall/CRC, New York, 1994.

[97] Halvor Eifring and Rolf Theil, *Linguistics for students of asian and african languages*, http ://www.uio.no/studier/emner/hf/ikos/EXFAC03-AAS/h05/larestoff/linguistics, 2005.

[98] D.M. Endres and J.E. Schindelin, *A new metric for probability distributions*, Information Theory, IEEE Transactions on **49** (2003), no. 7, 1858 – 1860.

[99] Seyda Ertekin, Jian Huang, Léon Bottou, and C. Lee Giles, *Learning on the border : Active learning in imbalanced data classification*, Proceedings of the 16th Conference on Information and KnowledgeManagement, CIKM2007 (Lisboa), ACM Press, November 2007.

[100] Tom Fawcett, *"in vivo" spam filtering : a challenge problem for KDD*, SIGKDD Explorations Newsletter **5** (2003), no. 2, 140–148.

[101] ———, *An introduction to ROC analysis*, Pattern Recogn. Lett. **27** (2006), no. 8, 861–874.

[102] F. Fdez-Riverola, E.L. Iglesias, F. Diaz, J.R. Mendez, and J.M. Corchado, *Applying lazy learning algorithms to tackle concept drift in spam filtering*, Expert Systems with Applications **33** (2007), no. 1, 36 – 48.

[103] N. Freed and N. Borenstein, *RFC 2045 - Multipurpose Internet Mail Extensions (MIME) Part One : Format of Internet Message Bodies*, IETF, 1996.

[104] Yoav Freund and Robert E. Schapire, *A short introduction to boosting*, In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1999, pp. 1401–1406.

[105] Jerome H. Friedman, *On bias, variance, 0/1—loss, and the curse-of-dimensionality*, Data Min. Knowl. Discov. **1** (1997), no. 1, 55–77.

[106] Sylvia Frühwirth-Schnatter, *Finite mixture and markov switching models*, 1st ed., Springer Series in Statistics, Springer, New York, NY 10013, USA, 2006.

[107] B. Fuglede and F. Topsoe, *Jensen-shannon divergence and hilbert space embedding*, june-2 july 2004, p. 31.

[108] K. Fukumizu, *Statistical active learning in multilayer perceptrons*, Neural Networks, IEEE Transactions on **11** (2000), no. 1, 17–26.

[109] Giorgio Fumera, Ignazio Pillai, and Fabio Roli, *Spam filtering based on the analysis of text information embedded into images*, Journal of Machine Learning Research **6** (2006), 2699–2720.

[110] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy, *Mining data streams : a review*, SIGMOD Rec. **34** (2005), no. 2, 18–26.

[111] Evgeniy Gabrilovich and Shaul Markovitch, *Text categorization with many redundant features : using aggressive feature selection to make svms competitive with c4.5*, ICML '04 : Proceedings of the twenty-first international conference on Machine learning (New York, NY, USA), ACM, 2004, p. 41.

[112] William Gale, *Good-turing smoothing without tears*, Journal of Quantitative Linguistics **2** (1994).

[113] Michael G. Gemignani, *Elementary topology*, 2nd ed., Dover Publications, Inc, Mineola, NY, 1990, reprint from 1972 edition.

[114] Joshua Goodman and Geoff Hulten, *Tutorial on junk email filtering*, http ://research.microsoft.com/ joshuago/tutorialOnJunkMailFilteringjune4.pdf, 2004.

[115] Joshua Goodman and Wen tau Yih, *Online discriminative spam filter training*, CEAS 2006 : Proceedings of the 3rd Conference on Email and Anti-Spam, 2006.

[116] P. Graham, *A plan for spam*, http ://www.paulgraham.com/spam.html, 2002.

[117] _____, *Better bayesian filtering*, http ://www.paulgraham.com/better.html, 2003.

[118] John Graham-Cumming, *SpamOrHam*, Virus Bulletin (2006-06-01).

[119] Martijn Grooten, *Vbspam comparative review*, Virus Bulletin (May 2010), 24–31.

[120] David A. Grossman and Ophir Frieder, *Information Retrieval : Algorithms and heuristics*, 2nd ed., Springer, Dordrecht, The Netherlands, 2004.

[121] Zhenmei Gu and N. Cercone, *Naive bayes modeling with proper smoothing for information extraction*, Fuzzy Systems, 2006 IEEE International Conference on, 2006, pp. 393–400.

[122] Dan Gusfield, *Algorithms on Strings, Trees and Sequences : Computer Science and Computational Biology*, Cambridge University Press, New York, NY, 1997.

[123] Thiago S. Guzella and Walmir M. Caminhas, *A review of machine learning approaches to spam filtering*, Expert Systems with Applications **36** (2009), no. 7, 10206 – 10222.

[124] James D. Hamilton, *Time series analysis*, Princeton University Press, Princeton, NJ, USA, 1994.

[125] David Hand, *Classifier technology and the illusion of progress*, Statistical Science **21** (2006), no. 1, 1–14.

[126] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning : Data mining, inference and prediction*, Springer, New York, 2001.

[127] Brian Hayes, *How many ways can you spell Viagra ?*, American Scientist **95** (2007).

[128] J. M. G. Hidalgo, *Evaluating cost-sensitive unsolicited bulk email categorization*, SAC '02 : Proceedings of the 2002 ACM Symposium on Applied Computing (Madrid), ACM Press, March 2002, pp. 615–620.

[129] Wen-Feng Hsiao and Te-Min Chang, *An incremental cluster-based approach to spam filtering*, Expert Syst. Appl. **34** (2008), no. 3, 1599–1608.

[130] MessageLabs Intelligenge, *Spammers become multilingual whilst web-malware writers take a break*, http ://www.messagelabs.com/mlireport/MLIReport_2009.07_July_FINAL.pdf, July 2009.

[131] Md. Rafiqul Islam, Morshed U. Chowdhury, and Wanlei Zhou, *An innovative spam filtering model based on support vector machine*, CIMCA '05 : Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation (Washington, DC, USA), IEEE Computer Society, 2005, pp. 348–353.

[132] Radwan Jalam, *Apprentissage automatique et catégorisation de textes multilangues*, Ph.D. thesis, Université Lumière Lyon 2, Lyon FR, Juin 2003.

[133] Klaus Jänich, *Topology*, Springer Verlag, New York NY, 1984.

[134] Nathalie Japkowicz and Mohak Shah, *Evaluating learning algorithms - a classification perspective*, Cambridge University Press, New York, NY, 2011.

[135] H. Jeffreys, *An invariant form for the prior probability in estimation problems*, Proceedings Royal Society (London - Series A) **186** (1946), 453–461.

[136] Thorsten Joachims, *Text categorization with support vector machines : Learning with many relevant features*, Springer Verlag, 1998, pp. 137–142.

[137] Thorsten Joachims, *Making large-scale support vector machine learning practical*, (1999), 169–184.

[138] ———, *Learning to classify text using support vector machines : Methods, theory and algorithms*, Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[139] Thorsten Joachims, *Training linear svms in linear time*, KDD '06 : Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (New York, NY, USA), ACM Press, 2006, pp. 217–226.

[140] George H. John and Pat Langley, *Estimating continuous distributions in bayesian classifiers*, Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 1995, pp. 338–345.

[141] Ioannis Kanaris, Konstantinos Kanaris, Ioannis Houvardas, and Efstathios Stamatatos, *Words versus character n-grams for anti-spam filtering*, International Journal on Artificial Intelligence Tools **16** (2007), no. 6, 1047–1067.

[142] Michael J. Kearns and Umesh V. Vazirani, *An introduction to computational learning theory*, The MIT Press, Cambridge, MA, USA, 1994.

[143] Mark G. Kelly, David J. Hand, and Niall M. Adams, *The impact of changing populations on classifier performance*, KDD '99 : Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining (New York, NY, USA), ACM, 1999, pp. 367–371.

[144] Vlado Keselj, Evangelos Milios, Andrew Tuttle, and Singer Wang, *DalTREC 2005 Spam Track : Spam filtering using N-gram-based techniques*, 2005.

[145] J. Kiefer and J. Wolfowitz, *Stochastic estimation to the maximum of a regression function*, Annals of Mathematical Statistics **23** (1952), no. 3, 462–466.

[146] Kazuaki Kishida, *Technical issues of cross-language information retrieval : a review*, Information Processing & Management **41** (2005), no. 3, 433 – 455, Cross-Language Information Retrieval.

[147] Zvi Kohavi, *Switching and finite automata theory*, 2nd ed., Tata McGraw Hill, New Delhi, 1978.

[148] A. Kolcz and J. Alspector, *SVM-based filtering of E-mail spam with content-specific misclassification costs*, TextDM 2001 (IEEE ICDM-2001 Workshop on Text Mining) (2001).

[149] Aleksander Kolcz, Michael Bond, and James Sargent, *The challenges of service-side personalized spam filtering : scalability and beyond*, InfoScale '06 : Proceedings of the 1st international conference on Scalable information systems (New York, NY, USA), ACM, 2006, p. 21.

[150] Aleksander Kolcz and Abdur Chowdhury, *Hardening fingerprinting by context*, CEAS, 2007.

[151] Aleksander Kolcz and Gordon V. Cormack, *Genre-based decomposition of email class noise*, KDD '09 : Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (New York, NY, USA), ACM, 2009, pp. 427–436.

[152] Daphne Koller and Nir Friedman, *Probabilistic graphical models*, The MIT Press, Cambridge, MA, USA, 2009.

[153] Alan G. Konheim, *Cryptography : A primer*, 1st ed., John Wiley & Sons, New York, NY, 1981.

[154] Sadanori Konishi and Genshiro Kitagawa, *Information criteria and statistical modeling*, Springer, New York, 2008.

[155] Igor Kononenko and Matjaz Kukar, *Machine Learning and Data Mining*, Horwood Publishing, Chichester, UK, 2007.

[156] S. Kullback, *Information theory and statistics*, 2nd ed., Dover Publications Inc., Mineola, NY, 1968.

[157] S. Kullback and R. A. Leibler, *On information and sufficiency*, Annals of Mathematical Statistics **22** (1951), 49–86.

[158] Ludmila I. Kuncheva, *Classifier ensembles for changing environments*, Multiple Classifier Systems (Fabio Roli, Josef Kittler, and Terry Windeatt, eds.), Lecture Notes in Computer Science, vol. 3077, Springer Berlin / Heidelberg, 2004, 10.1007/978-3-540-25966-4_1, pp. 1–15.

[159] Harold J. Kushner and G. George Yin, *Stochastic approximation and recursive algorithms and applications*, 2nd ed., Stochastic Modelling and Applied Probability, vol. 35, Springer-Verlag, New York, NY, 2003.

[160] John Lafferty and Chengxiang Zhai, *Document language models, query models, and risk minimization for information retrieval*, SIGIR '01 : Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 2001, pp. 111–119.

[161] Pat Langley and Wayne Iba, *Average-case analysis of a nearest neighbor algorithm*, Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (pp. 889–894). Chambery, Morgan Kaufmann, 1993, pp. 889–894.

[162] Pat Langley and Stephane Sage, *Scaling to domains with irrelevant features*, Computational Learning Theory and Natural Learning Systems (R. Greiner, ed.), vol. IV : Making Learning Systems Practical, MIT Press, Cambridge, MA, USA, 1997, pp. 51–63.

[163] Zhang Le, Tianshun Yao, and Yao Tian-shun, *Filtering junk mail with a maximum entropy model*, Proceeding of 20th International Conference on Computer Processing of Oriental Languages (ICCPOL03), 2003.

[164] L. Lebart and A. Salem, *Statistique textuelle*, Dunod, 1994.

[165] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998), no. 11, 2278 –2324.

[166] Yann LeCun, Leon Bottou, Genevieve Orr, and Klaus Müller, *Efficient backprop*, Neural Networks : Tricks of the Trade, Lecture Notes in Computer Science, 1998, p. 546.

[167] Honglak Lee and Andrey Y. Ng, *Spam deobfuscation using a hidden Markov model*, CEAS 2005 – The Second Conference on Email and Anti-Spam, 2005.

[168] E. L. Lehmann and Joseph P. Romano, *Testing statistical hypotheses*, 3rd ed., Springer Texts in Statistics, Springer, New York, NY, 2005.

[169] Edda Leopold and Jörg Kindermann, *Text categorization with support vector machines. how to represent texts in input space ?*, Mach. Learn. **46** (2002), no. 1-3, 423–444.

[170] David D. Lewis, *An evaluation of phrasal and clustered representations on a text categorization task*, SIGIR '92 : Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 1992, pp. 37–50.

[171] ———, *Feature selection and feature extraction for text categorization*, HLT '91 : Proceedings of the workshop on Speech and Natural Language (Morristown, NJ, USA), Association for Computational Linguistics, 1992, pp. 212–217.

[172] David D. Lewis and William A. Gale, *A sequential algorithm for training text classifiers*, SIGIR '94 : Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), Springer-Verlag New York, Inc., 1994, pp. 3–12.

[173] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li, *RCV1 : A New Benchmark Collection for Text Categorization Research*, J. Mach. Learn. Res. **5** (2004), 361–397.

[174] F. Liese and I. Vajda, *On divergences and informations in statistics and information theory*, Information Theory, IEEE Transactions on **52** (2006), no. 10, 4394 –4412.

[175] J. Lin, *Divergence measures based on the shannon entropy*, Information Theory, IEEE Transactions on **37** (1991), no. 1, 145–151.

[176] J. K. Lindsey, *Statistical analysis of stochastic processes in time*, Cambridge Series in Statistics and Probability Mathematics, Cambridge University Press, Cambridge, UK, 2004.

[177] Patrick Lindstrom, Sarah Jane Delany, and Brian Mac Namee, *Autopilot : simulating changing concepts in real data*, 19th. Irish Conference on Artificial Intelligence and Cognitive Science, Dublin Institute of Technology, 2008.

[178] Greg Louis, *Greg's bogofilter page - tuning bogofilter*, http ://www.bgl.nu/bogofilter/, 2004.

[179] Daniel Lowd and Christopher Meek, *Adversarial learning*, KDD '05 : Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (New York, NY, USA), ACM, 2005, pp. 641–647.

[180] _____, *Good word attacks on statistical spam filters*, Proc. CEAS 2005 – Second Conference on Email and Anti-Spam (Palo Alto, CA), 2005.

[181] Thomas R. Lynam and Gordon V. Cormack, *On-line spam filter fusion*, SIGIR '06 : Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 2006, pp. 123–130.

[182] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schuze, *Introduction do information retrieval*, Cambridge University Press, New York, NY, USA, 2008.

[183] Christopher D. Manning and Hinrich Schuze, *Foundations of statistical natural language processing*, MIT Press, Cambridge, MA, USA, 1999.

[184] J. Mayfield and P. McNamee, *Indexing using both n-grams and words*, 7th Text REtrieval Conference (Gaithersburg, MD), 1998, pp. 230–243.

[185] Andrew McCallum and Kamal Nigam, *A comparison of event models for naive bayes text classification*, AAAI-98 Workshop on Learning for Text Categorization, 1998.

[186] Edward M. McCreight, *A space-economical suffix tree construction algorithm*, J. ACM **23** (1976), no. 2, 262–272.

[187] Geoffrey McLachlan and David Peel, *Finite mixture models*, 1st ed., Wiley Series in Probability and Statistics, John Wiley & Sons Inc, New York, NY, USA, 2000.

[188] V. Metsis, I. Androutsopoulos, and G. Paliouras, *Naive Bayes – which naive Bayes ?*, Proc. CEAS 2006 – Third Conference on Email and Anti-Spam (Mountain View, CA), 2006.

[189] Marvin L. Minsky and Seymour A. Papert, *Perceptrons, an Introduction to Computational Geometry*, The MIT Press, December 1969.

[190] Tom Mitchell, *Machine learning*, McGraw-Hill, 1997.

[191] Claudiu Musat and George Petre, *On the relevance of spam feeds*, Virus Bulletin (October 2010), 21–24.

[192] Gonzalo Navarro and Veli Mäkinen, *Compressed full-text indexes*, ACM Comput. Surv. **39** (2007), no. 1, 2.

[193] Günter Neumann and Sven Schmeier, *Combining shallow text processing and machine learning in real world applications*, In Proceedings of the IJCAI-99 workshop on Machine Learning for Information Filtering, 1999.

[194] Ngo Phuong Nhung and Tu Minh Phuong, *An efficient method for filtering image-based spam e-mail*, CAIP, 2007, pp. 945–953.

[195] Cormac O'Brien and Carl Vogel, *Spam filters : Bayes vs. chi-squared ; letters vs. words*, ISICT '03 : Proceedings of the 1st international symposium on Information and communication technologies, Trinity College Dublin, 2003, pp. 291–296.

[196] F. Osterreicher and I. Vajda, *Statistical information and discrimination*, Information Theory, IEEE Transactions on **39** (1993), no. 3, 1036–1039.

[197] F. Österreicher and I. Vajda, *A new class of metric divergences on probability spaces and its applicability in statistics*, Annals of the Institute of Statistical Mathematics **55** (2003), no. 3, 639–653.

[198] Levent Özgür, Tunga Güngör, and Fikret Gürgen, *Adaptive anti-spam filtering for agglutinative languages : a special case for turkish*, Pattern Recogn. Lett. **25** (2004), no. 16, 1819–1831.

[199] R. M. Pampapathi, B. Mirkin, and M. Levene, *A suffix tree approach to email filtering*, Tech. report, Birkbeck University of London, 2005.

[200] P. Pantel and D. Lin, *Spamcop : A spam classification and organization program*, Learning for Text Categorization : Papers from the 1998 Workshop (Madison, Wisconsin), AAAI Technical Report WS-98-05, 1998, pp. 95–98.

[201] M. del C. Pardo and I. Vajda, *On asymptotic properties of information-theoretic divergences*, Information Theory, IEEE Transactions on **49** (2003), no. 7, 1860 – 1867.

[202] Robert K. Plice, Nigel P. Melville, and Oleg V. Pavlov, *Toward an information-compatible anti-spam strategy*, Commun. ACM **52** (2009), no. 5, 128–130.

[203] M. F. Porter, *An algorithm for suffix stripping*, Readings in information retrieval (1997), 313–316.

[204] Calton Pu and Steve Webb, *Observed trends in spam construction techniques*, Proc. CEAS 2006 – Third Conference on Email and Anti-Spam (Mountain View, CA), 2006.

[205] R Development Core Team, *R : A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2010, ISBN 3-900051-07-0.

[206] Jason D. M. Rennie, *ifile : An application of machine learning to mail filtering*, Proceedings of the KDD-2000 Workshop on Text Mining, 2000.

[207] P. Resnick, *RFC 2822 - Internet Message Format*, IETF, 2001.

[208] L. Rigutini, M. Maggini, and Bing Liu, *An em based training algorithm for cross-language text categorization*, Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on, Sept. 2005, pp. 529–535.

[209] B.D. Ripley, *Pattern recognition and neural networks*, Cambridge University Press, Cambridge U.K., 1996.

[210] H. Robbins and S. Monro, *A stochastic approximation method*, Annals of Mathematical Statistics **22** (1951), no. 3, 400–407.

[211] Christian P. Robert, *The bayesian choice*, 2nd ed., Springer Texts in Statistics, Springer, New York, NY, 2007.

[212] Christian P. Robert and George Casella, *Monte carlo statistical methods*, 2nd ed., Springer Texts in Statistics, Springer, New York, NY, 2004.

[213] S. E. Robertson and K. Sparck Jones, *Simple, proven approaches to text retrieval*, Tech. Report 356, University of Cambridge – Computer Laboratory, 1997.

[214] G. Robinson, *Gary Robinson's spam rants*, http ://radio.weblogs.com/0101454/categories/spam/, 2004.

[215] Monica Rogati and Yiming Yang, *High-performing feature selection for text classification*, CIKM '02 : Proceedings of the eleventh international conference on Information and knowledge management (New York, NY, USA), ACM, 2002, pp. 659–661.

[216] F. Rosenblatt, *Principles of neurodynamics*, Spartan Books, New York, NY, 1962.

[217] Frank Rosenblatt, *The perceptron : A probabilistic model for information storage and organization in the brain*, Psychological Review **65** (1958), no. 6, 386–408.

[218] R. Rosenfeld, *Two decades of statistical language modeling : where do we go from here ?*, Proceedings of the IEEE **88** (2000), no. 8, 1270–1278.

[219] David Saad (ed.), *Online learning and neural networks*, Cambridge University Press, Cambridge, UK, 1998.

[220] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz, *A bayesian approach to filtering junk E-mail*, Learning for Text Categorization : Papers from the 1998 Workshop (Madison, Wisconsin), AAAI Technical Report WS-98-05, 1998.

[221] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos, *A memory-based approach to anti-spam filtering for mailing lists*, Inf. Retr. **6** (2003), no. 1, 49–73.

[222] G. Salton, A. Wong, and C. S. Yang, *A vector space model for automatic indexing*, Commun. ACM **18** (1975), no. 11, 613–620.

[223] M. Sasaki and H. Shinnou, *Spam detection using text clustering*, CW '05 : Proceedings of the 2005 International Conference on Cyberworlds (Washington, DC, USA), IEEE Computer Society, 2005, pp. 316–319.

[224] K. M. Schneider, *A comparison of event models for naive bayes anti-spam e-mail filtering*, EACL '03 : Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics (Morristown, NJ, USA), Association for Computational Linguistics, 2003, pp. 307–314.

[225] Bernard Schölkopf and Alexander J. Smola, *Learning with kernels : Support vector machines, regularization and beyond*, MIT Press, Cambridge, MA, USA, 2002.

[226] D. Sculley, *Advances on online learning-based spam filters*, Ph.D. thesis, Tufts University, 2008.

[227] ———, *Going mini : Extreme lightweight spam filters*, Proc. CEAS 2009 – Sixth Conference on Email and Anti-Spam (Mountain View, CA), 2009.

[228] D. Sculley and G. V. Cormack, *Filtering spam in the presence of noisy user feedback*, Proceedings of the 5th Conference on Email and Anti-Spam (CEAS 2008), 2008.

[229] D. Sculley and Gabriel M. Wachman, *Relaxed online SVMs for spam filtering*, SIGIR '07 : Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 2007, pp. 415–422.

[230] D. Sculley and Gabriel M. Wachman, *Relaxed online SVMs in the TREC Spam Filtering Track*, Sixteenth Text REtrieval Conference (TREC-2007) (Gaithersburg, MD), NIST, 2007.

[231] Fabrizio Sebastiani, *Machine learning in automated text categorization*, ACM Computing Surveys **34** (2002), no. 1, 1–47.

[232] R. Segal, J. Crawford, J. Kephart, and B. Leiba, *SpamGuru : An enterprise anti-spam filtering system*, First Conference on Email and Anti-Spam (CEAS), 2004.

[233] Richard Segal, *Combining global and personal anti-spam filtering*, CEAS 2007 - The Fourth Conference on Email and Anti-Spam, 2-3 August 2007, Mountain View, California, USA, 2007.

[234] Burr Settles, *Active learning literature survey*, Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[235] H. S. Seung, M. Opper, and H. Sompolinsky, *Query by committee*, COLT '92 : Proceedings of the fifth annual workshop on Computational learning theory (New York, NY, USA), ACM, 1992, pp. 287–294.

[236] Claude E. Shannon, *A mathematical theory of communication*, Bell System Technical Journal **27** (1948), 379–423, 625–56.

[237] ———, *Communication theory of secrecy systems*, Bell Systems Technical Journal **28** (1949), no. 4, 656–671.

[238] ———, *Prediction and entropy of printed english*, Bell Systems Technical Journal **30** (1951), 50–64.

[239] John Shawe-Taylor and Nello Cristianini, *Kernel methods for pattern analysis*, Cambridge University Press, 2004.

[240] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Faith Cranor, Jason Hong, and Chengshan Zhang, *An empirical analysis of phishing blacklists*, Proc. CEAS 2009 – Sixth Conference on Email and Anti-Spam (Mountain View, CA), 2009.

[241] H. Shimodaira, *Improving predictive inference under covariate shift by weighting the log-likelihood function*, Journal of Statistical Planning and Inference **90** (2000), no. 2, 227–244.

[242] C. Siefkes, F. Assis, S. Chhabra, and W. S. Yerazunis, *Combining winnow and orthogonal sparse bigrams for incremental spam filtering*, PKDD '04 : Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (New York, NY, USA), Springer-Verlag New York, Inc., 2004, pp. 410–421.

[243] Amit Singhal, Gerard Salton, Mandar Mitra, and Chris Buckley, *Document length normalization*, Information Processing & Management **32** (1996), no. 5, 619 – 633.

[244] David F. Skoll, *Rptn - a mechanism for sharing bayes votes*, http ://www.roaringpenguin.com/files/rptn.pdf, May 2005.

[245] Steve Smale and Yuan Yao, *Online learning algorithms*, Found. Comput. Math. **6** (2006), no. 2, 145–170.

[246] Amos Storkey, *Dataset shift in machine learning*, Neural Information Processing, ch. When Training and Test Sets Are Different, Characterizing Learning Transfer, pp. 3–28, The MIT Press, Cambridge, Massachussets, February 2009.

[247] Ching Y. Suen, *n-gram statistics for natural language understanding and text processing*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **PAMI-1** (1979), no. 2, 164–172.

[248] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller, *Covariate shift adaptation by importance weighted cross validation*, J. Mach. Learn. Res. **8** (2007), 985–1005.

[249] Wen tau Yih, Robert McCann, and Aleksander Kołcz, *Improving spam filtering by detecting gray mail*, Proc. CEAS 2007 – Fourth Conference on Email and Anti-Spam (Mountain View, CA), 2007.

[250] Chris Thornton and Bn Qh, *Separability is a learner's best friend*, Proceedings of the Fourth Neural Computation and Psychology Workshop : Connectionist Representations, Springer-Verlag, 1997, pp. 40–47.

[251] Simon Tong and Daphne Koller, *Support vector machine active learning with applications to text classification*, Journal of Machine Learning Research **2** (2002), 45–66.

[252] F. Topsoe, *Some inequalities for information divergence and related measures of discrimination*, Information Theory, IEEE Transactions on **46** (2000), no. 4, 1602 –1609.

[253] Alexey Tsymbal, *The problem of concept drift : Definitions and related work*.

[254] Esko Ukkonen, *On-line construction of suffix trees*, Algorithmica **14** (1995), no. 3, 249–260.

[255] Vladimir N. Vapnik, *The nature of statistical learning theory*, 2nd ed., Springer, New York, NY, 1995.

[256] ———, *Statistical learning theory*, 1st ed., John Wiley & Sons, New York, NY, 1998.

[257] Jinlong Wang, Ke Gao, Yang Jiao, and Gang Li, *Study on ensemble classification methods towards spam filtering*, Advanced Data Mining and Applications, Springer, 2009, pp. 314–325.

[258] Peng Wang, Haixun Wang, Xiaochen Wu, Wei Wang, and Baile Shi, *A low-granularity classifier for data streams with concept drifts and biased class distribution*, IEEE Trans. on Knowl. and Data Eng. **19** (2007), no. 9, 1202–1213.

[259] Zhe Wang, William Josephson, Qin LV, Moses Charikar, and Kai Li, *Filtering image spam with near-duplicate detection*, CEAS 2007 – The Third Conference on Email and Anti-Spam, 2007.

[260] Larry Wasserman, *All of statistics - a concise course in statistical inference*, 1st ed., Springer Texts in Statistics, Springer, New York, NY, 2004.

[261] Ming wei Chang, Wen tau Yih, and Robert Mccann, *Personalized spam filtering for gray mail*, In Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS, 2008.

[262] Gerhard Widmer and Miroslav Kubat, *Learning in the presence of concept drift and hidden contexts*, Machine Learning **23** (1996), 69–101, 10.1007/BF00116900.

[263] Bernard Widrow and Marcian E. Hoff, *Adaptive switching circuits*, Neurocomputing : Foundations of Research, pp. 123–134, MIT Press, Cambridge, MA, USA, 1988.

[264] Andrew K. C. Wong and Manlai You, *Entropy and distance of random graphs with application to structural pattern recognition*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **PAMI-7** (1985), no. 5, 599 –609.

[265] Chunyu Yang and Jie Zhou, *Non-stationary data sequence classification using online class priors estimation*, Pattern Recogn. **41** (2008), no. 8, 2656–2664.

[266] Yiming Yang and Jan O. Pedersen, *A comparative study on feature selection in text categorization*, ICML '97 : Proceedings of the Fourteenth International Conference on Machine Learning (San Francisco, CA, USA), Morgan Kaufmann Publishers Inc., 1997, pp. 412–420.

[267] William S. Yerazunis, *Correspondence with Paul Graham*, http ://www.paulgraham.com/wsy.html, 16 October, 2002.

[268] ———, *Sparse binary polynomial hashing and the CRM114 discriminator*, http ://-crm114.sourceforge.net/docs/CRM114_paper.html, 2003.

[269] ———, *The spam-filtering accuracy plateau at 99.9% accuracy and how to get past it*, 2004 MIT Spam Conference, January 2004.

[270] W. Yih, J. Goodman, and G. Hulten, *Learning at low false positive rates*, CEAS 2006 : Proceedings of the 3rd Conference on Email and Anti-Spam, 2006.

[271] S.M. Yiu, P.Y. Chan, T.W. Lam, W.K. Sung, H.F. Ting, and P.W.H. Wong, *Allowing mismatches in anchors and whole genome alignement*, WSEAS Transactions on Biology and Biomedicine, vol. 4, World Scientific and Engineering Academy and Society, January 2007, pp. 1–6.

[272] Marco Zaffalon and Marcus Hutter, *Robust feature selection by mutual information distributions*, Proceedings of the 18th International Conference on Uncertainty in Artificial Intelligence (UAI-2002, Morgan Kaufmann, 2002, pp. 577–584.

[273] Jonathan A. Zdziarski, *Ending spam - bayesian content filtering and the art of statistical language classification*, No Starck Press, 2005.

[274] Le Zhang, Jingbo Zhu, and Tianshun Yao, *An evaluation of statistical spam filtering techniques*, ACM Transactions on Asian Language Information Processing (TALIP) **3** (2004), no. 4, 243–269.

[275] Yan Zhou, Zach Jorgensen, and Meador Inge, *Combating good word attacks on statistical spam filters with multiple instance learning*, Proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007) (Los Alamitos, CA, USA), vol. 2, IEEE Computer Society, 2007, pp. 298–305.

[276] Djamel A. Zighed, Stéphane Lallich, and Fabrice Muhlenbach, *Principles of data mining and knowledge discovery*, vol. 2431, ch. Separability Index in Supervised Learning, pp. 241–267, Springer, Berlin, 2002.

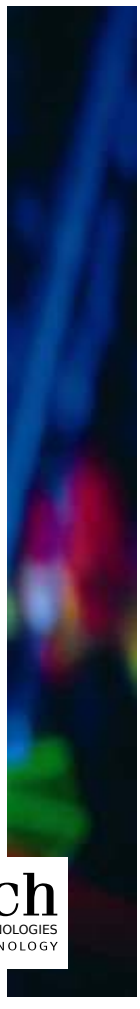
Contribution au classement statistique mutualisé de messages électroniques *(spam)*

Résumé : Depuis la fin des années 90, les différentes méthodes issues de l'apprentissage artificiel ont été étudiées et appliquées au problème de classement de messages électroniques (*filtrage de spam*), avec des résultats très bons, mais pas parfaits. Il a toujours été considéré que ces méthodes étaient adaptées aux solutions de filtrage orientées vers un seul destinataire et non pas au classement des messages d'une communauté entière.
Dans cette thèse notre démarche a été, d'abord, de chercher à mieux comprendre les caractéristiques des données manipulées, à l'aide de corpus réels de messages, avant de proposer des nouveaux algorithmes. Puis, nous avons utilisé un simple classificateur discriminant linéaire avec de l'apprentissage actif en ligne - pour démontrer empiriquement qu'avec un algorithme simple et une configuration d'apprentissage adaptée au contexte réel de classement, on peut obtenir des résultats aussi bons que ceux que l'on obtient avec des algorithmes plus complexes. Nous avons aussi démontré, avec des ensembles de messages d'un petit groupe d'utilisateurs, que la perte d'efficacité peut ne pas être significative dans un contexte de classement mutualisé.
Mots clés : spam, classement de messages électroniques, filtrage de spam, filtrage mutualisé, apprentissage actif, apprentissage en ligne.

A contribution to shared classification of electronic messages *(spam)*

Abstract: Since the 90's, different machine learning methods were investigated and applied to the email classification problem (*spam filtering*), with very good but not perfect results. It was always considered that these methods are well adapted to filter messages to a single user and not filter to messages of a large set of users, like a community.
Our approach was, at first, look for a better understanding of handled data, with the help of a corpus of real messages, before studying new algorithms. With the help of a simple linear discriminator classifier with online active learning, we could show empirically that with a simple classification algorithm coupled with a learning strategy well adapted to the real context it's possible to get results which are as good as those we can get with more complex algorithms. We also show, empirically, with the help of messages from a small group of users, that the efficiency loss is not very high when the classifier is shared by a group of users.

Keywords: spam, email classification, spam filtering, shared filtering, active learning, online learning.



MINES ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY