Classification automatique de textes dans des catégories non thématiques

Romain Vinot

Thèse

Présentée pour obtenir le grade de
Docteur de l'Ecole Nationale Supérieure des Télécommunications
(Spécialité : Informatique et Réseaux)

Romain VINOT

Classification automatique de textes dans des catégories non thématiques

soutenue le 9 février 2004 devant un jury composé de :

Ludovic Lebart (GET-ENST)	Président du jury
Florence d'Alché-Buc (LIP6)	Rapporteurs
Martin Rajman (EPFL)	
Yannick Toussaint (LORIA)	Examinateurs
Éric Gaussier (Xerox)	
François Yvon (GET-ENST)	Directeur de thèse

Remerciement

Ce document n'aurait pas vu le jour sans la présence et le soutien d'un certain nombre de personnes qui ont contribué, de près ou de loin, à l'aboutissement du travail qu'il représente. La première personne concernée est bien entendu mon directeur de thèse, François Yvon, pour ses conseils et ses critiques constructives qui m'ont aidé et stimulé tout au long de ces années.

Je suis redevable aux membres de mon jury : Ludovic Lebart qui a bien voulu présider la soutenance, Florence d'Alché-Buc et Martin Rajman pour avoir accepter d'être rapporteur de mon travail, ainsi que Yannick Toussaint et Éric Gaussier pour leur participation en tant qu'examinateur.

Cette thèse a été réalisée en collaboration avec la société Akio Solutions et le département Informatique et Réseaux de l'École Nationale Supérieure des Télécommunications. Je tiens à remercier Kha Tran pour la position qu'il a pu me fournir pendant la première année et surtout Michel Riguidel pour le soutien financier qu'il a pu m'offrir pendant les deux années suivantes.

Je tiens à exprimer également mon amitié à toute l'équipe d'Akio, tout particulièrement Guillaume Schmid, Philippe Duperron, Christophe Musielak et Jean-Martial Le Magorou pour le travail que nous avons réalisé et les bons moments que nous avons passé ensemble (pour une petite séance de kart, c'est quand vous voulez).

De la même façon, j'exprime ma reconnaissance à l'ensemble des membres du département Informatique et Réseaux qui ont égayé mon séjour en ces lieux : Jean-Philippe Demoulin et Nicolas Stroppa pour avoir partager mon bureau toutes ces années et m'avoir fait découvert la bière bien fraiche, Nicolas Saunier pour animer le bureau de musique, Cyril Carrez et son vélo unijambiste, mais aussi tous les autres membres du département Jean-Louis et Laleh pour leur thé régulier, Alain pour ses blagues et ses proverbes, Olivier et Irène pour leurs dégustations de fromage, Nadine pour m'avoir accompagné les premiers mois, ainsi que Jean-Marc, Talel, Alda, Sam, Alexandre, Pascal, Jean et Antoine en m'excusant auprès de ceux que je ne cite pas.

Pour finir, je remercie tous ceux qui, en dehors du travail, m'ont accompagné et soutenu durant ces trois années. Je réserve une pensée particulière pour Cécile qui traverse la vie avec moi pour les bons et les mauvais moments.

Résumé

Résumé court

La classification automatique de textes était jusqu'à présent employée pour l'indexation documentaire. À travers quatre exemples, nous présentons quelques caractéristiques de nouveaux contextes applicatifs ainsi que leurs conséquences pour les algorithmes existants. Nous mettons ainsi en évidence le fait que Rocchio, d'ordinaire peu performant, est particulièrement adapté aux corpus bruités et à une utilisation semi-automatique mais très désavantagé avec des classes définies par plusieurs thèmes. Nous proposons une extension de Rocchio, Rocchio Multi-Prototypes, pour gérer les classes multi-thématiques en adaptant la complexité de son modèle d'apprentissage. RMP utilise un algorithme de classification faiblement supervisée qui détecte des sous-classes et sélectionne les plus utiles pour la catégorisation. Nous proposons aussi un algorithme de détection de changements de concept dans des corpus à flux temporel à partir du calcul du taux d'activité des sous-classes.

Automatic text categorization was used up to now for Document Indexing. With four examples, we show some caracteristics of new applications with their consequences for some existing text classifiers. We highlight the fact that Rocchio, usually not very accurate, is very well adapted to noisy corpora and semi-automatic uses but performs poorly when classes contain many different topics. We propose an extension of Rocchio, Rocchio Multi-Prototypes, to deal with multi-topical classes by adapting its model complexity. RMP uses a weakly supervised clustering algorithm which detects sub-classes and keeps the most useful ones for the categorization. We propose also an algorithm to detect concept shifting in temporal corpora by the use of an activity ratio for each sub-class.

Résumé long

La classification automatique de textes était jusqu'ici surtout employée pour indexer des documents suivant leur sujet principal. La technologie commence aujourd'hui à être déployée hors du domaine des documentalistes : par exemple dans le cadre de la lutte contre les courriers électroniques non sollicités ou dans le domaine du Knowledge Management et du Customer Relationship Management. Le contexte de ces nouvelles applications est parfois radicalement différent de celui pour lequel les techniques ont été développées initialement.

Nous commençons cette thèse par une analyse qualitative de quatre corpus qui correspondent à différentes applications : le filtrage de courriers électroniques non sollicités, la détection des opinions exprimées dans un document, la sélection d'une réponse à une question dans le cadre du Mail Center et le classement de dépêches suivant l'évènement générateur. Nous mettons en évidence plusieurs caractéristiques qui les relient. Premièrement, les classes ne se définissent pas par un thème précis. On peut parfois les décrire comme un regroupement de plusieurs sous-thèmes. Deuxièmement, le corpus est souvent constitué avec moins de soins que pour les applications classiques : les documents sont mal étiquetés et non représentatifs des futurs textes à classer. Troisièmement, les classifieurs sont souvent utilisés pour du routage en mode semi-automatique : le classifieur ne prend pas de décision, il ne fait que proposer les catégories à un expert humain qui les valide ou non. Ce mode d'exploitation modifie profondément la façon de mesurer la satisfaction des utilisateurs.

À la suite de la mise en évidence de ces caractéristiques, nous avons analysé le comportement de quelques algorithmes de catégorisation par rapport à ces spécificités. Nous avons ainsi mis en évidence expérimentalement et après une explication théorique que l'algorithme Rocchio, d'ordinaire considéré comme peu performant, est particulièrement adapté aux corpus très bruités et aux mesures de performance reflétant un mode d'utilisation semi-automatique.

Dans une deuxième partie, nous montrons que Rocchio est désavantagé dans le cas où les classes sont constituées de plusieurs sous-thèmes très dispersés. Nous proposons alors une modification pour résoudre ce problème. Le principe de Rocchio Multi-Prototypes (RMP) est de détecter les sous-classes dispersées à l'aide d'un algorithme de clustering faiblement supervisé et d'appliquer ensuite Rocchio à ces sous-classes. L'intérêt principal de notre approche provient de l'aspect faiblement supervisé de l'algorithme qui parvient (i) à détecter le nombre optimal de clusters et (ii) à sélectionner les clusters les plus utiles pour la classification. Nous avons comparé notre algorithme à d'autres classifieurs standards sur plusieurs corpus artificiels et réels. Les résultats montrent que RMP est significativement supérieur à Rocchio dans les cas où les classes sont très dispersées et identique à ce dernier dans les cas plus classiques.

Dans une troisième partie, nous montrons que notre algorithme de clustering peut s'avérer utile pour d'autres tâches que la simple catégorisation. Nous présentons l'exemple de la détection de changements de concepts sur les corpus à flux temporel. Notre méthode s'appuie sur l'attribution d'un poids à chaque cluster directement lié à la date des documents qui le composent. Nous avons montré qu'il est légèrement moins réactif que les algorithmes standards de détection de changements de concepts lorsque le changement est très brusque mais qu'il est capable de détecter les changements même dans un environnement très variable où de nombreux concepts apparaissent et disparaissent régulièrement (contrairement aux algorithmes existants), ce qui est souvent le cas dans les nouvelles applications que nous avons étudiées.

Table des matières

Chapitre

1

Introduction

1.1 Contexte

1.1.1 Un rapide tour historique de la catégorisation de textes

Le domaine du traitement intelligent de données textuelles regroupe tous les outils et les méthodes capables d'extraire des informations de textes écrits dans une langue naturelle. Au vu de la quantité phénoménale de textes aujourd'hui accessibles (par le biais du réseau Internet en particulier), il devient urgent de trouver des solutions opérationnelles pour automatiser le plus grand nombre possible de tâches liées à ces documents.

Il existe essentiellement deux domaines de recherche qui abordent cette problématique. Les méthodes issues de la statistique et de l'analyse de données cherchent surtout à proposer des outils aux statisticiens et aux linguistes pour leur permettre d'analyser de grands volumes de données en fournissant des informations synthétiques sur les corpus. Le but n'est pas ici de fournir un système de traitement automatique mais plutôt un ensemble d'outils que les experts peuvent utiliser pour découvrir des concordances ou des tendances générales difficilement décelables à la main. On trouve dans cette catégorie les logiciels d'analyse de corpus qui fournissent des listes de fréquences de mots et des représentations graphiques souvent issues des techniques d'analyse factorielle des correspondances.

La deuxième approche consiste à proposer des systèmes de type « boîte noire » qui traitent les données textuelles de façon automatique sans intervention humaine. Les fonctions réalisées sont souvent de bas niveau (analyse lexicale, analyse syntaxique

de surface, recherche d'informations par mots-clés) ou très spécialisées. Les moteurs de recherche, qui ont été popularisés par l'avènement du réseau Internet, fournissent l'exemple d'une application héritée de cette approche. La communauté de Recherche d'Informations (RI) explore depuis vingt ans de nouveaux modèles de tâches susceptibles de correspondre à de nouvelles préoccupations des utilisateurs. La tâche de filtrage propose ainsi de modéliser les intérêts à long terme des utilisateurs sous la forme d'une requête de RI qui peut perdurer plusieurs mois ou même sans horizon précis. Seuls les documents qui semblent pertinents pour la requête sont envoyés à l'utilisateur.

La Catégorisation de Textes (CT) est une tâche générique qui englobe le filtrage. Elle consiste à assigner une ou plusieurs catégories parmi une liste prédéfinie à un document. Ces catégories correspondent dans la plupart des cas au sujet principal du texte. Cette classification permet par exemple de structurer l'accès aux ouvrages d'une bibliothèque en rangeant les documents par thème. L'application la plus emblématique de cette approche est la catégorisation des dépêches d'agence de presse suivant leur thème (par exemple économie, politique, sport).

1.1.2 La situation actuelle

La CT est aujourd'hui un domaine de recherche bien établi et très actif. Les travaux portent depuis une quinzaine d'années sur les systèmes avec apprentissage des catégories à partir de corpus pré-étiquetés. Plusieurs modèles de tâches ont été spécifiés comme le filtrage (classification supervisée bi-classe), le routage (classification supervisée multi-classe) ou le classement ordonné (listage des documents par ordre de pertinence pour chaque catégorie). Les méthodologies de tests et les infrastructures de campagnes d'évaluation ont été mises en place avec succès. Les pré-traitements et méthodes de numérisation des textes sont maintenant bien connus. Il existe de multiples algorithmes de classification qui fonctionnent correctement mais déterminer les avantages des uns par rapport aux autres reste souvent délicat.

Ces systèmes ont été appliqués avec succès à de nombreux domaines. Nous avons déjà parlé de la catégorisation des dépêches d'agence de presse. Les utilisateurs peuvent spécifier les thèmes qui les intéressent. Chaque dépêche est catégorisée suivant son thème principal et est envoyée à tous les utilisateurs qui ont indiqué un intérêt pour ce thème. La CT a également été mise en oeuvre pour le classement d'articles scientifiques. Ce classement permet (i) d'appréhender rapidement le contenu du corpus en lisant la liste des thèmes et (ii) de retrouver rapidement tous les articles parlant d'un même sujet. Les mêmes motivations sont valables pour le classement de brevets. Une autre application où la CT est particulièrement utile est le classement de sites web dans un annuaire en ligne de type Yahoo. Le point commun entre toutes ces applications est la quantité importante de documents. La CT permet aux utilisateurs d'appréhender plus facilement

cette masse d'information.

1.1.3 La problématique

Étant donnée la dématérialisation des communications et des données en général, la quantité d'informations à traiter augmente très rapidement. De nombreuses applications sont aujourd'hui demandeuses de la technologie de CT. C'est le cas de façon exemplaire des courriers électroniques. La quantité de courriers reçus augmente régulièrement au fur et à mesure que de plus en plus de personnes utilisent cette méthode de communication. Nous sommes actuellement à un point où certains utilisateurs n'ont matériellement plus le temps de répondre à leur courrier. Beaucoup de logiciels souhaitent utiliser les méthodes de CT pour fournir automatiquement des informations supplémentaires sur chaque email : le courrier est-il une publicité, est-il urgent, à quel sujet peut-il être rattaché ? C'est également le cas de façon plus globale de la gestion des documents dans l'entreprise. L'idée du Knowledge Management (KM) est de rationaliser toute cette gestion en automatisant le plus de tâches possibles. À nouveau, la CT a un grand rôle à jouer dans ce type d'applications.

Avec ces nouvelles problématiques, la CT sort du domaine de la gestion purement documentaliste des textes. Il ne s'agit plus ici de détecter le sujet d'un document afin de mettre de l'ordre dans une bibliothèque numérique. Les finalités de l'outil sont ici bien différentes : il s'agit de traiter automatiquement un flux textuel pour effectuer une action sur chaque document. Les catégories que l'on cherche à différencier ne correspondent plus nécessairement au sujet principal du document mais doivent s'appuyer sur l'organisation du travail et la répartition des tâches dans l'entreprise.

Nous savons que les performances des classifieurs, comme tous les algorithmes d'apprentissage, dépendent énormément des corpus sur lesquels elles sont mesurées. Pourtant, à l'heure actuelle, il n'existe pas d'étude qui compare l'utilisation classique de la CT sur des catégories thématiques et son utilisation dans le contexte des nouvelles applications comme le Knowledge Management. Il est donc difficile pour un industriel de savoir quel algorithme choisir, dans quel contexte, et à quelles performances il peut s'attendre. Le manque de garantie sur les résultats d'un procédé qui, par ailleurs, est intrinsèquement imparfait, empêche la CT d'être utilisée à grande échelle (d'autres problèmes comme les formats de fichiers ou la confidentialité des données sont également un frein à l'adoption de ces méthodes).

1.2 Contributions de ce travail

Les contributions de cette thèse sont de trois ordres : une analyse non exhaustive de l'applicabilité de la CT dans quelques contextes non standards, la mise au point

d'une sur-couche aux algorithmes de CT afin de mieux prendre en compte l'aspect non nécessairement thématique des catégories et mise à l'essai systématique des divers algorithmes présentés sur de nombreux corpus réels .

Notre première contribution concerne l'utilisation effective de la CT : nous montrons que les performances varient beaucoup suivant les applications. Le caractère non thématique des catégories n'est pas forcément un problème pour les classifieurs. Lorsque les catégories sont facilement repérables par quelques mots très discriminants, les algorithmes de CT sont directement utilisables. Lorsque la tâche de catégorisation est plus difficile et nécessite une compréhension plus poussée du message, beaucoup de facteurs rentrent en ligne de compte. Quelques-uns d'entre eux sont mis en évidence au cours de cette thèse.

Notre deuxième contribution a été la mise au point d'un algorithme pour détecter les structures thématiques à l'intérieur de chaque classe. Cet algorithme permet ainsi d'obtenir des informations supplémentaires sur le corpus qui peuvent être utiles à plus d'un titre. Nous montrons deux applications de cet algorithme : (i) utilisé comme une sur-couche des algorithmes de classification, il permet d'améliorer leur performance dans le cas où les catégories ne sont pas thématiques ; (ii) dans un contexte d'évolution rapide des concepts sous-jacents aux catégories, il permet de détecter les changements de concepts pour adapter le corpus d'apprentissage rapidement.

Un troisième point important qui a guidé notre travail est l'importance que nous avons donné à l'implémentation. Ceci se traduit par un souci constant d'expérimenter toutes les techniques proposées sur le plus grand nombre possible de corpus et s'est conclu par l'intégration d'un module de CT dans le logiciel Akio Mail Center[1]. Quelques détails sur l'implémentation sont donnés dans l'annexe A.

1.3 Plan du document

Cette thèse est constituée de 5 chapitres principaux ainsi qu'une introduction et une conclusion.

Le chapitre 2 est consacré à l'état de l'art de la Catégorisation de Textes. Nous présentons au départ le modèle formel de la classification ainsi que les diverses variantes de cette tâche générique. Puis nous passons en revue toute la chaîne de traitement d'un système de CT. La description de la numérisation des documents comprend le modèle du *sac de mots* (*bag of words*) dans lequel chaque document est représenté par la liste non ordonnée des mots qu'il contient, les méthodes supplémentaires de transformations des mots en attributs plus précis et moins ambigües, et enfin les calculs de pondérations pour faire ressortir les attributs potentiellement discriminants. Nous

[1] http ://www.akio-software.com

passons ensuite aux algorithmes de classification en séparant clairement ceux issus de la Recherche d'Informations et ceux issus de l'Apprentissage Automatique. Nous détaillons plus particulièrement les trois algorithmes que nous utilisons dans le reste de la thèse : Rocchio, k-PPV et SVM. Pour finir, nous indiquons les différentes méthodes existantes pour mesurer la qualité des systèmes et les comparer.

Le chapitre 3 présente un panorama de quatre applications de la CT dans lesquelles les systèmes sont utilisés dans un contexte non standard avec des catégories non thématiques. Pour le problème du courrier électronique non sollicité (Spam), il est aujourd'hui reconnu que les techniques de CT permettent d'obtenir un système adaptatif très performant qui surpasse souvent les autres méthodes existantes. Ce domaine est donc l'un des premiers à voir arriver la CT dans les produits commerciaux. Le problème de la détection de points de vue est présenté via le projet Princip. Il s'agit, dans ce projet, de différencier parmi les pages parlant de racisme, celles dont les opinions exprimées sont anti-racistes de celles qui sont pro-racistes. La complexité de cette application ne semble pas résider dans l'aspect non-thématique des catégories mais plutôt dans la difficulté à sélectionner les fragments pertinents pour la catégorisation. La troisième application que nous avons étudiée concerne le contexte du Mail Center et la possibilité de sélectionner automatiquement une réponse parmi un ensemble de réponses-types à un courrier électronique contenant une question. Il est possible d'avoir plusieurs réponses-types concernant un même thème. La catégorisation n'est donc plus thématique. Cette problématique semble très difficile, mais l'apport d'un module de CT, même imparfait, permet des gains importants de productivité. Enfin, la dernière application concerne le classement des dépêches suivant l'évènement qu'elles relatent. Bien que non thématique (la catégorie ne définit pas un thème mais un évènement, il peut donc y avoir deux évènements portant sur le même thème mais correspondant à deux catégories différentes), cette problématique est très simple car la présence d'un ou deux mots spécifiques suffit souvent à discriminer sans erreur les documents.

Dans le chapitre 4, nous revenons sur les quatre exemples d'applications pour analyser plus précisément les caractéristiques importantes qui rendent le problème différent de la classification thématique standard. Nous commençons d'abord par préciser le modèle de tâche qui semble prépondérant dans ces nouvelles problématiques. Dans ces applications, l'affectation dans une catégorie conditionne généralement le fait d'effectuer une action sur le document, le mode d'utilisation est donc plus proche du routage que du filtrage. Nous montrons que, pour le routage, deux techniques pourtant communément utilisées (racinisation des mots et Query Zoning), ne sont pas nécessaires et peuvent même dégrader les performances. Nous analysons ensuite plus en détail les caractéristiques de ces applications. Nous montrons ainsi que la CT peut être employée même lorsque les résultats sont imparfaits, à condition qu'elle soit utilisée en mode semi-automatique,

c'est-à-dire avec une validation humaine du résultat. Dans ce cas, les mesures de performance qui simulent la satisfaction des utilisateurs doivent être transposées à ce nouveau contexte, modifiant ainsi les mérites respectifs des différents algorithmes. Les classifieurs qui optimisent explicitement les mesures classiques sont moins à l'aise en mode semi-automatique que les modèles simples. Une autre caractéristique importante concerne le fort niveau de bruit des corpus dans les nouvelles applications. À nouveau, les algorithmes qui optimisent précisément la classification sont souvent gênés par un niveau de bruit important alors que les algorithmes moins spécialisés sont plus robustes. À la suite de ces analyses, nous arrivons à la conclusion que les algorithmes génératifs simples (comme Rocchio ou Naïve Bayes) ont des performances relativement stables. D'autres algorithmes qui optimisent directement les mesures de performance ont des comportements plus fluctuants dès que la tâche est légèrement modifiée ou que les corpus ne sont pas tout à fait comme dans les applications classiques.

Dans le chapitre 5, nous attaquons enfin le problème des catégories multi-thématiques. À la suite des analyses réalisées aux chapitres 3 et 4, nous avons pu mettre en évidence que les catégories peuvent en général être définies non pas par un thème principal mais par un ensemble de thèmes, le sujet principal de chaque document étant alors un des thèmes de la classe. Avec une représentation lexicale classique des textes, nous avons montré que, si certains classifieurs sont peu gênés par cette modification, les algorithmes génératifs simples qui représentent une classe par un profil unique sont particulièrement désavantagés lorsque les classes sont complexes. Nous présentons dans ce chapitre un algorithme de clustering dont le but est de détecter qu'une classe est formée de plusieurs thèmes et de la diviser en plusieurs clusters. L'apport principal de cet algorithme par rapport à d'autres travaux du même type est de proposer un algorithme qui utilise les étiquettes de classe (algorithme faiblement supervisé) afin de construire des clusters uniquement dans le cas où c'est utile et de conserver les classes intactes dans le cas contraire. Cet algorithme est ensuite appliqué comme pré-traitement à Rocchio et SVM. Nous mettons en évidence qu'il permet effectivement de trouver le nombre optimal de clusters et qu'il est particulièrement utile avec Rocchio pour certaines applications comme le Mail Center.

Cette méthode de classification non supervisée a servi dans le chapitre 5 à améliorer les performances des algorithmes de classification. Le clustering fournit une information structurelle sur la présence ou l'absence de multiples sous-thèmes dans une classe. Nous pensons que cette information peut être utile dans d'autres contextes que la catégorisation. Nous présentons dans le chapitre 6 l'utilisation du clustering pour la détection de changements de concepts dans un flux de documents. Notre méthode se contente de calculer un taux d'activité pour chaque cluster, qui joue le même rôle que la probabilité d'apparition dans un futur proche d'un document aux alentours du cluster. Tous les

travaux que nous avons pu trouver sur ce problème se contentent de détecter qu'il y a eu un changement. L'emploi du clustering nous permet aussi de détecter le concept qui a changé et donc de ne supprimer que les documents qui sont devenus non pertinents au lieu de supprimer tous les documents anciens sans distinction. Nous avons réalisé des expériences sur des corpus artificiels pour analyser le comportement de notre algorithme et le comparer à d'autres travaux. Nous avons également, et ceci pour la première fois à notre connaissance [2], testé notre algorithme de détection de concepts sur un corpus réel, ce qui nous a permis de mettre en évidence son intérêt par rapport à l'utilisation d'une simple fenêtre temporelle. Nous avons ainsi pu mettre en évidence que notre méthode est capable de traiter les corpus très fluctuants, pour lesquels des changements de concepts apparaissent très fréquemment contrairement à la technique de fenêtre adaptative qui ne fonctionne pas bien dans ce contexte.

Enfin, nous concluons cette thèse au chapitre 7 en résumant les diverses contributions que nous avons pu apporter et en présentant les perspectives de recherche de nos travaux.

[2] Il existe déjà des travaux avec corpus réels sur l'obsolescence des classes mais la problématique semble légèrement différente du problème de détection des changements à l'intérieur des classes.

CHAPITRE

2

La catégorisation de textes

Sommaire du chapitre

LA *catégorisation de textes* (CT) est le processus qui consiste à assigner une ou plusieurs catégories parmi une liste prédéfinie à un document. Ce principe général peut être appliqué à de nombreuses situations : organisation des ouvrages dans une bibliothèque (les documentalistes parlent aussi d'*indexation* : il s'agit de déterminer dans quel rayon les documents doivent être placés, les catégories sont donc les différents rayons), de sites web dans un annuaire en ligne (les catégories sont ici les différentes

rubriques, les sites étant souvent organisés avec une page web par rubrique), d'une base de données de textes spécialisés (brevets ou publications scientifiques par exemple).

Pour effectuer manuellement cette tâche de classification, il faut lire attentivement chaque texte. Cette opération prend énormément de temps et est donc extrêmement coûteuse. L'idée de la faire effectuer automatiquement par des machines remonte aux années 60. Mais c'est l'explosion de la quantité des documents électroniques et l'incapacité de les traiter manuellement qui a dynamisé la recherche dans ce domaine depuis une quinzaine d'années.

Au début des années 90, les travaux proviennent essentiellement de la communauté de *Recherche d'Informations* (RI). Les méthodes de numérisation de texte (le *modèle vectoriel*), les algorithmes de classification et les méthodologies de test ont été adaptés à la CT en particulier au cours des conférences TREC [NDa]. La communauté d'*Apprentissage Automatique* (AA) s'est interessée à ce problème il y a une dizaine d'années en le considérant comme un domaine d'application pour ses algorithmes de reconnaissance des formes. Actuellement, les méthodes de numérisation de textes restent largement inspirées de la RI alors que les classifieurs les plus performants sont issus de l'AA.

Il existe une autre communauté qui traite également de la problématique de la classification textuelle. Elle est composée essentiellement de statisticiens et de linguistes et s'appuie surtout sur les méthodes d'analyse de données. Dans ce cadre, le but n'est pas de créer un système qui classe automatiquement des documents sans intervention humaine mais d'extraire des informations synthétiques d'un corpus. L'ordinateur n'est ici qu'un outil qui fournit aux experts des données à interpréter. Les problématiques étudiées sont par exemple l'étude des genres littéraires [IJ01] ou la détermination de l'auteur d'un texte [Mul64]. Nous ne traiterons pas ce type de problématiques dans cette thèse.

La CT provient de plusieurs domaines scientifiques différents qui n'utilisent pas toujours le même vocabulaire, en particulier pour la dénomination des différentes tâches. Il est nécessaire de bien clarifier les termes que nous allons employer. La *catégorisation* correspond à la *classification supervisée* pour l'AA et à la *discrimination* en statistiques alors que la RI emploie des termes plus proches de l'application concernée : *filtrage* ou *routage*. La *classification non supervisée* de l'AA correspond en statistiques à la *classification* ou *clustering*, qui est également le terme utilisé en RI. Pour éviter toute confusion, nous emploierons les termes catégorisation et clustering.

Dans ce chapitre, nous allons décrire les différents modules qui composent un système de CT. Nous commencerons par une description formelle de la tâche de catégorisation (section 2.1), puis nous préciserons comment les documents textuels sont représentés (section 2.2) avant d'être transmis aux différents algorithmes d'apprentis-

sage (sections 2.3,2.4 et 2.5). Enfin, nous présenterons les différents moyens mis à notre disposition pour comparer les systèmes de CT (section 2.6). La lecture de ce chapitre peut être complétée par celle de l'article de Fabrizio Sebastiani : Machine Learning in Automated Text Categorization [Seb02].

2.1 Description de la catégorisation de texte

Nous nous plaçons, pour le moment, dans le cadre restreint où un document ne peut appartenir qu'à une seule catégorie. Nous verrons dans le paragraphe suivant comment généraliser à plusieurs classes. Soit $\mathcal{C} = \{c_1, ..., c_n\}$ un ensemble de *catégories* ou de *classes* et $\mathcal{D} = \{d_1, ..., d_N\}$ un ensemble de documents. À chaque document d_i est associée une classe c_i à laquelle il appartient. L'objectif est d'apprendre une fonction $f : \mathcal{D} \rightarrow \mathcal{C}$ telle que pour tout couple $\langle d_i, c_i \rangle$, $f(d_i) = c_i$. La fonction f est appelée *classifieur*. Cette fonction peut être construite manuellement ou automatiquement à partir d'un ensemble de couples document-catégorie.

Dans le cas manuel comme le système CONSTRUE [HANS90], le principe est de faire analyser le corpus par des experts qui fournissent des règles qui déterminent une catégorie à partir de la présence d'un ou deux mots spécifiques. Ces règles sont de la forme :« si le mot xxx est présent en même temps que le mot yyy, alors la classe est zzz ». Dans le cas automatique, c'est un algorithme qui analyse le corpus. La fonction calculée f n'est pas toujours intelligible par des humains car elle peut faire intervenir un grand nombre de valeurs numériques qu'un humain ne peut pas appréhender. La détermination de cette fonction f est appelée *phase d'apprentissage*. f est ensuite utilisée pour attribuer une catégorie à tout nouveau document. C'est la *phase d'exploitation* ou *phase de test*. Les méthodes sans apprentissage sont très coûteuses à développer (car elles nécessitent un travail important de la part d'experts humains). Elles sont aujourd'hui peu développées et la majorité des travaux portent surtout sur les méthodes avec apprentissage. Nous nous intéressons dans cette thèse uniquement à ces dernières.

Suivant les dénominations données par les conférences TREC, nous parlerons de *filtrage* lorsqu'un document peut être attribué à plusieurs classes. C'est le cas classique de la catégorisation de dépêches d'agence de presse suivant leur thème (économie, politique, sport). Une dépêche ayant plusieurs thèmes peut donc se retrouver dans plusieurs catégories. Lorsqu'un document ne peut être attribué qu'à une seule classe, on parle de *routage* (par exemple lorsqu'il s'agit de déterminer la personne la plus compétente pour répondre à un courrier électronique).

Dans la formalisation précédente, un document ne peut être affecté qu'à une et une seule catégorie. Dans le cas où la tâche nécessite qu'un document puisse être affecté à plusieurs classes, il faut alors transformer le problème d'origine à n classes en de nombreux sous-problèmes à deux classes. De la façon la plus générale une classification

en n classes non-exclusives peut se formaliser comme une classification en 2^n classes exclusives correspondant aux 2^n sous-ensembles possibles de l'ensemble des n classes. Comme 2^n est généralement trop grand, on fait alors l'hypothèse d'indépendance des classes qui permet de se ramener à n problèmes. Pour chaque catégorie c_j, on construit un nouveau problème opposant la classe des documents pertinents (appartenant à la catégorie c_j) à la classe des documents non-pertinents (n'appartenant pas à la catégorie c_j).

Par ailleurs, certains algorithmes d'apprentissage ne sont pas capables de prendre en compte plus de deux classes. Cela n'est pas gênant dans le cas du filtrage puisque la tâche se transforme naturellement en plusieurs sous-problèmes bi-classe. Dans le cas du routage en revanche, le passage à plusieurs sous-problèmes bi-classe sans perte d'information n'est pas facile. Comme les classes sont exclusives, les décisions ne sont pas indépendantes. Les solutions les plus utilisées sont les suivantes : *un contre tous* et *un contre un*. Dans le premier cas, on construit un sous-problème pour chaque classe en mettant dans une classe les éléments d'une catégorie et dans l'autre tous les documents appartenant aux autres catégories. La décision finale est d'attribuer le document à la classe avec le score le plus élevé. Dans le second cas, on construit $\frac{n(n+1)}{2}$ sous-problèmes correspondant à toutes les paires possibles de classes. Pour catégoriser un document, un tournoi est réalisé et la classe finale est attribuée à celle qui a gagné le plus de matchs (principe du *winner takes all*) ou à celle qui a le taux de confiance moyen le plus élevé [Fri96, HT98]. Bien que la méthode "un contre un" soit théoriquement plus performante, ce résultat n'est pas toujours vérifié expérimentalement [ASS00]. D'autres méthodes plus évoluées faisant appel aux codes correcteurs d'erreurs ont également été proposées [DB95, HPRZ02, Ber99, Gha01].

Il existe encore d'autres types de tâches moins utilisés, que nous ne traiterons pas ici, par exemple le *ranking* (où il s'agit de fournir une liste de documents par ordre de pertinence décroissante). On se rapproche ici de la RI vue comme un problème de catégorisation (c'est en fait de cette formulation qu'est née la CT) : il s'agit de déterminer pour chaque document s'il appartient à la classe "pertinent" ou à la classe "non pertinent" pour une requête. Le corpus d'apprentissage est alors constitué uniquement de la requête et éventuellement des documents fournis par l'utilisateur lors du feedback (voir section 2.3.1).

2.2 Représentation des documents

La quasi-totalité des systèmes de CT représentent les documents par la présence/absence de *termes* dans le texte. Ces termes sont les unités minimales constitutives d'un texte, ils peuvent être plus ou moins complexes : chaînes de caractères, mots, racines de mots, groupes de mots ou expressions. Dans tous les cas, la séquentialité des mots dans le

document est irrémédiablement perdue et on ne retient que le nombre d'occurrences du terme. C'est la métaphore du *sac de mots* [SM83]. Un document d_i est alors représenté par un vecteur d'attributs $\mathbf{d_i} = (d_{i1}, ..., d_{iV})$, où d_{ik} représente "l'importance" du terme k dans le document d_i et V est la taille du vocabulaire. Chaque document est donc représenté par un vecteur numérique dans un espace de très grande dimension.

L'objectif recherchée à travers cette représentation est de décrire le contenu thématique des documents (lorsque la catégorisation n'est pas thématique, d'autres principes de sélection d'attributs sont mis en oeuvre [IJ01, Mul64]). Chaque attribut doit donc correspondre à un trait sémantique. Dans la majorité des systèmes, les termes correspondent aux mots car ils sont faciles à segmenter et à détecter à partir du texte brut et ils forment une relativement bonne approximation des concepts sémantiques. Néanmoins, le sens des mots reste ambigu dans de nombreux cas (polysémie, contexte) et plusieurs mots peuvent correspondre au même concept (synonymie). Pour éviter ces difficultés inhérentes aux langues naturelles, des travaux ont pris comme unité minimale non pas des mots mais des groupes de mots. Pour cela, il est nécessaire d'utiliser des techniques de statistique et de Traitement Automatique des Langues Naturelles (TALN) pour cibler les groupes de mots syntaxiquement cohérents et potentiellement intéressants [Bes02, BMP02, LC90]. Dans l'état actuel des choses, on obtient par ce moyen une amélioration très faible qui ne justifie pas la lourdeur des calculs supplémentaires. D'après [Lew92], même si les groupes de mots (mots composés, expressions ou autres) ont généralement une signification sémantique moins ambiguë, ils apparaissent moins souvent que les mots simples. Les fréquences d'apparition associées sont donc statistiquement moins fiables et les estimations qui en découlent ne sont pas assez précises.

2.2.1 Pré-traitements inspirés de la RI

Dans tous les cas (utilisation de mots simples ou de groupes de mots), il est préférable d'effectuer quelques pré-traitements afin de filtrer les mots non informatifs et de regrouper les mots de même famille.

Une première opération consiste à supprimer les mots faisant partie d'une liste prédéfinie : les *stopwords*. Ce sont des mots génériques non porteurs de sens tels que articles, déterminants, auxiliaires qui sont a priori inutiles pour discriminer les différentes catégories thématiques. Ils peuvent donc être supprimés sans perte d'information utile. On sait expérimentalement que le nombre d'occurrences de mots dans un corpus suit approximativement une loi de Zipf [Mil58]. Cette loi stipule que le nombre d'occurrences d'un mot M varie comme une puissance du rang i de ce mot (chaque mot étant rangé suivant leur nombre d'occurrences) :

$$M_i = \frac{M_1}{i^a} \tag{2.1}$$

avec l'exposant a très proche de 1. Cette loi rend compte du fait que quelques mots apparaissent très souvent alors que beaucoup apparaissent très rarement. Les stopwords sont généralement les mots qui apparaissent le plus souvent dans un texte. Ils sont peu nombreux (entre une centaine et un millier suivant les listes) mais comme ils sont fréquents, leur suppression permet de diminuer significativement le nombre total d'occurrences dans le corpus. Cette liste doit être dressée préalablement pour chaque langue.

Une deuxième opération consiste à conserver, non pas les mots eux-mêmes, mais leur racine ou leur lemme (c'est-à-dire l'entrée-dictionnaire d'un terme). Ce principe permet : (i) de prendre en compte les variations flexionnelles (singulier/pluriel, conjugaison, ...) ou dérivationnelles (substantifs, verbes adjectivés, ...) en regroupant sous le même terme tous les mots de la même famille (dont on suppose qu'ils portent le même sens et qu'ils appartiennent au même thème) et donc d'améliorer la classification en mettant en relation des documents qui n'ont pourtant pas de mot en commun et en ayant des termes plus génériques dont l'estimation des occurrences sera plus fiable ; (ii) de baisser le nombre total de termes et donc les temps de calcul. La méthode exacte consiste à faire de la lemmatisation en utilisant des outils de TALN pour trouver le lemme de chaque mot. Ceci nécessite beaucoup de ressources linguistiques (dictionnaires, règles de dérivations, etc). De plus les résultats contiennent encore des erreurs à cause des problèmes d'ambiguité et de l'impossibilité d'avoir des dictionnaires complets. La méthode de *racinisation* (ou stemming) [Por80] utilise des heuristiques simples à partir de règles de remplacement de chaînes de caractères pour supprimer les suffixes les plus utilisés. Sans dictionnaire, elle ne gère que les règles principales et ne peut pas prendre en compte les nombreuses exceptions des règles de dérivations. Par exemple, en français, l'une des règles stipule de supprimer le "e" final de chaque mot. Le mot *fraise* est alors transformé en *frais*, ce qui induit une relation entre les deux termes qui n'existe pas. La méthode a également le défaut de fournir un résultat sous la forme d'une "pseudo-racine" qui ne correspond à aucune réalité linguistique (le terme *catégorisation* devient *catégoris* qui n'est pas un mot existant). Il est alors difficile de faire des analyses linguistiques après une phase de racinisation. En contrepartie, elle est bien plus simple à mettre en oeuvre que la lemmatisation. Et les mots mal racinisés restent relativement rares (les expériences montrent que dans le cadre de la RI la racinisation est aussi performante que la lemmatisation). Ces deux techniques ont surtout été étudiées dans le cadre de l'anglais qui est une langue avec très peu de flexions. Les résultats obtenus doivent être re-validés sur d'autres langues, en particulier sur celles ayant une morphologie très différente.

En RI, la requête d'origine est souvent très courte et ne contient que quelques mots-clés. De plus, aucune information de supervision sur le corpus ne permet de déterminer l'utilité relative des différents mots du vocabulaire. Ces deux opérations, suppression des stopwords et racinisation (ou lemmatisation), permettent d'améliorer sensiblement

les performances des systèmes. La suppression des stopwords permet de supprimer une information abondante mais non pertinente et donc de se focaliser sur les éléments plus utiles de la requête. La racinisation permet de mettre en relation des documents pertinents même s'ils n'ont pas de termes en commun (mais qu'ils partagent des mots de la même famille). Pour la CT, en revanche, le corpus d'apprentissage est beaucoup plus grand. Les algorithmes d'apprentissage peuvent, au cours de la phase d'apprentissage, détecter l'inutilité des mots non informatifs, et par conséquent non discriminants, et leur attribuer un poids très faible. La suppression préalable des stopwords est donc inutile puisque cette opération est déjà réalisée implicitement pendant l'apprentissage. La mise en correspondance de mots de la même famille comme le permet la racinisation n'est pas utile en CT. Le corpus étant très important, il est possible de ne s'intéresser qu'aux documents qui contiennent exactement les mots recherchés et obtenir néanmoins un nombre conséquent de documents plutôt que d'élargir la recherche aux mots de la même famille. La racinisation est donc également moins utile en CT. Ces deux techniques peuvent même avoir une influence négative comme le montre Riloff [Ril95], qui indique que les informations qu'elles suppriment sont potentiellement utiles. En revanche, elles permettent de diminuer fortement les temps de calcul (les stopwords étant les plus fréquents, leur suppression élimine beaucoup d'occurrences dans le corpus, et la racinisation permet, dans le cas de l'anglais, de réduire en général d'un tiers la taille du vocabulaire).

2.2.2 Réduction de dimensionnalité

Malgré ces premières transformations , la taille du vocabulaire peut encore s'avérer très importante (de quelques dizaines de milliers à plusieurs centaines de milliers de termes). Il est souvent préférable de le diminuer davantage avant d'appliquer les techniques de classification les plus complexes. Ceci permet, d'une part, de rendre les temps de calcul non prohibitifs et, d'autre part, d'améliorer les performances en diminuant le sur-apprentissage qui a tendance à apparaître lorsque le nombre de paramètres est trop important par rapport aux données [KS96]. Le classifieur est alors trop focalisé sur les exemples d'apprentissage et il n'est plus capable de généraliser correctement pour classer de nouveaux documents inconnus. Le sur-apprentissage dépend aussi beaucoup du classifieur utilisé : certains sont capables de sélectionner les termes utiles et ne sont pas gênés par une surabondance d'informations non pertinentes alors que d'autres considèrent que tous les attributs sont pertinents, il faut alors impérativement effectuer une sélection auparavant.

Le principe le plus utilisé consiste à calculer pour chaque terme une statistique qui représente son utilité pour la classification puis à sélectionner les x termes les plus importants. Il existe de nombreuses statistiques pour mesurer cette quantité d'information

à partir du nombre d'occurrences du terme dans la classe et hors de la classe. Ces statistiques s'interprètent généralement comme l'estimation d'une probabilité dans un modèle probabiliste ; Le tableau 2.1 présente les principales formules utilisées dont les performances sont comparées dans plusieurs études [YP97, GSS00, Mla98a, FH01, Spi00]. La *Fréquence-document* (Document Frequency) est très simple puisqu'elle correspond simplement au pourcentage de documents dans lesquels le terme apparaît (cette formule conduit à supprimer les termes qui apparaissent rarement). Le *Gain d'Information* mesure le nombre de bits nécessaires pour coder la catégorie d'un document selon que l'on fournit la valeur de l'attribut ou non. L'*Information Mutuelle* est utilisée généralement pour modéliser la corrélation entre deux variables aléatoires, ici le terme et la catégorie. Le *Chi-deux*(χ^2) mesure le degré d'indépendance de deux variables aléatoires. Si la *Fréquence-document* est légèrement moins performante que les autres avec de forts taux de sélection, son calcul est immédiat et obtient des performances équivalentes aux autres formules jusqu'à une réduction de $1/10^{\text{ème}}$ [YP97].

Nom	**Formule**
Fréquence-document	$P(t_k)$
Gain d'Information	$P(t_k, c_j) \log \frac{P(t_k,c_j)}{P(t_k)P(c_j)} + P(t_k, c_j) \log \frac{P(\overline{t_k},c_j)}{P(\overline{t_k})P(c_j)}$
Information Mutuelle	$\log \frac{P(t_k,c_j)}{P(t_k)P(c_j)}$
Chi-deux	$\frac{\lvert N\rvert[P(t_k,c_j)P(\overline{t_k},\overline{c_j})-P(t_k,\overline{c_j})P(\overline{t_k},c_j)]^2}{P(t_k)P(\overline{t_k})P(c_j)P(\overline{c_j})}$

TAB. 2.1 – Différentes mesures de la quantité d'informations contenus dans le terme t_k pour la classe c_j. $P(t_k, c_j)$ est la probabilité pour un document donné que le terme t_k soit présent et que le document soit de classe c_j. Elle est estimée par le rapport du nombre de documents de c_j dans lequel t_k apparaît sur le nombre total de documents. $\overline{A}$ indique l'absence de l'évènement A. $P(\overline{t_k}, \overline{c_j})$ est donc la probabilité que le terme t_k soit absent et que le document ne soit pas de la classe c_j (et ainsi de suite pour les autres probabilités).

Une façon alternative de diminuer le nombre de termes consiste à modifier la représentation des documents de telle sorte que les attributs ne correspondent plus à un terme simple mais à une combinaison de termes. On cherche souvent dans ces représentations à ce qu'un attribut corresponde à un concept sémantique. Cela va donc au-delà de la racinisation qui ne cherche qu'à regrouper les mots de même famille et non pas les mots de même sens. Dans le *term clustering* ([BM98a, ST01, SC01, KH00]), plusieurs termes sont regroupés pour former un nouvel attribut. Chaque attribut est donc censé représenter un concept sémantique, les différents termes étant les activateurs de ce concept. Le fait que plusieurs termes puissent activer un même concept permet de gérer la synonymie. La polysémie des mots est également prise en compte en permettant à un terme d'appartenir à plusieurs groupes. Pour déterminer ces groupes, il faut utiliser un

algorithme de clustering avec une mesure de similarité entre mots à partir des profils d'occurrence des termes dans les différentes classes.

S. Deerwester et S. Dumais [DDL+90] ont proposé le *Latent Semantic Analysis* (LSA), qui utilise la décomposition en valeurs singulières [LMP97] de la matrice d'occurrence $[X_{ik}]$ (x_{ik} est le nombre d'occurrences du terme k dans le document i). X se décompose en un produit de trois matrices $X = T*S*D^t$ avec S une matrice diagonale, T et D orthonormales. Les éléments de la matrice S sont les valeurs singulières de X (qui sont les racines carrés des valeurs propres de X^tX). Géométriquement, la matrice X représente les vecteurs documents avec une base canonique (à chaque terme correspond un axe). La décomposition revient à changer la représentation par un changement de base. Chaque terme est désormais représentée par une combinaison linéaire d'attributs. On utilise ensuite cette décomposition pour ne conserver que les k axes de plus fortes valeurs singulières. $\tilde{X} = T_k*S_k*D_k^t$ (S_k est la matrice diagonale issue de S en mettant à 0 tous les éléments d'indice supérieur à k) est une approximation de X qui est optimale au sens des moindres carrés (de toutes les matrices Y de rang k, $\tilde{X}$ est celle qui minimise la norme euclidienne de $(X - Y)$). C'est la réduction de dimensionnalité qui conserve le maximum d'informations. En revanche, la décomposition de la matrice est un processus lourd et cette méthode est donc très coûteuse en temps de calcul. De nombreux travaux ont expérimenté cette technique [SHP95] et ont étendu son utilisation comme [Hof00] qui en fournit une version probabiliste, [AL01] qui généralise le principe et propose un nouvel algorithme ou encore [ZH01] qui y incorpore des données hétérogènes.

Ces deux techniques, Term Clustering et LSA, semblent améliorer les performances de quelques pourcents. En revanche, elles nécessitent un temps de calcul important pendant l'apprentissage et il n'existe pas de méthode incrémentale qui permettrait de ne pas refaire tout le processus à chaque ajout de nouveaux documents. Il n'existe pas non plus, à notre connaissance, d'études concernant la stabilité des clusters. En ce qui concerne LSA, nous savons que les sous-espaces générés par plusieurs vecteurs propres (par exemple les k premiers) sont stables, mais que les vecteurs propres pris indivuellement sont très instables [SS90]. L'utilisation de ces deux techniques dépend donc du contexte de l'application et de la vitesse des changements susceptibles d'intervenir dans le corpus.

2.2.3 Calcul des poids

Une fois la liste des attributs déterminée, il reste à donner une pondération à chacun d'entre eux. Le choix le plus simple est d'utiliser une pondération binaire : 1 si le terme est présent dans le document, 0 dans le cas contraire. Beaucoup de travaux en RI dans les années 80-90 ont eu pour objectif d'améliorer les pondérations pour que les termes les plus significatifs aient un poids relativement plus élevé. Le processus a été

essentiellement empirique à partir de grandes campagnes d'évaluation où de nombreuses pondérations étaient comparées les unes aux autres (voir par exemple les conférences TREC [NDa]). Lorsque les algorithmes d'AA ont commencé à être utilisés en CT, les documents étaient souvent représentés avec le modèle binaire simple parce qu'on supposait que les classifieurs seraient capables de trier eux-mêmes les termes importants ou parce que l'algorithme nécessitait une représentation symbolique binaire. Mais comme pour la RI, l'utilisation de poids calculés à partir de formules spécifiques au domaine textuel permet généralement d'améliorer les performances.

Il existe essentiellement deux méthodes de calcul. Le *modèle vectoriel* de G. Salton développé au sein du logiciel SMART [SWY75, Sal71] est fondé sur des heuristiques qui se sont affinées au cours des dernières décennies. Le *modèle probabiliste* est utilisé lorsque les pondérations utilisées dans le classifieur ont une interprétation probabiliste.

Le modèle vectoriel

Le modèle vectoriel comprend deux composantes : le calcul du poids proprement dit et son utilisation dans les mesures de similarité entre documents. Dans ce modèle, les termes sont implicitement considérés comme indépendants (le calcul du poids d'un terme ne dépend pas de celui d'un autre terme). Bien que linguistiquement erronée, cette hypothèse ne semble pas être réellement un obstacle à l'utilisation du modèle. Domingos [DP96] montre que les performances du classifieur probabiliste Naïve Bayes (voir section 2.3.2) ne sont pas affectées par la violation de l'hypothèse d'indépendance des termes. Nous pensons que ceci est vrai pour beaucoup d'autres algorithmes qui font également cette hypothèse d'indépendance, en particulier tous les classifieurs qui utilisent le modèle vectoriel pour représenter les documents.

La pondération s'appuie sur deux notions : la *Fréquence du Terme* (ou *Term Frequency* : TF) qui prend en compte le nombre d'occurrences du terme dans le document et l'*Inverse de la Fréquence en Document* (ou *Inverse Document Frequency* : IDF) qui prend en compte le nombre d'occurrences du terme dans le corpus. Ces deux notions sont combinées multiplicativement de façon à attribuer un poids d'autant plus fort que le terme apparaît souvent dans le document et rarement dans le corpus complet. Il existe de multiples variantes du TF. Il peut être égal au nombre d'occurrences du terme, à son log, au log de son log, etc. L'utilisation du logarithme permet de diminuer l'importance des termes fortement répétés. On sait en effet que, en raison d'un effet de répétition dans les textes, plus un mot apparaît dans un document, plus il a de chances de réapparaître dans la suite de ce texte [Kat96]. Un terme prend davantage d'importance s'il passe de 1 à 2 occurrences que s'il passe de 10 à 11. Contrairement au TF, un large consensus existe pour l'IDF. Quelques travaux ont tenté d'expliquer pourquoi il fonctionne aussi bien : K. Papineni [Pap01] montre que la formule de l'IDF est la pondération

optimale lorsqu'il faut retrouver un document à partir d'une requête constituée de ce même document ; K. Church et B. Gale [CG95] rapprochent l'utilisation de IDF à une distribution poissonienne des termes dans un document. Pour l'ensemble des travaux de cette thèse, nous avons choisi la pondération *ltc* de SMART [Sal71] qui est l'une des plus performantes et a été la plus probante lors de nos expérimentations préalables.

$$d_{ik} = TFIDF(i,k) = TF(i,k) * IDF(k) = \log(1 + Occ(i,k)) * \log(\frac{N}{N_k}) \tag{2.2}$$

Avec $Occ(i,k)$ le nombre d'occurrences du terme k dans le document i, N le nombre total de documents et N_k le nombre de documents dans lequel apparaît le terme k. Avec cette pondération, plus un document est long (et contient par conséquence beaucoup d'occurrences), plus les poids d_{ik} augmentent. Pour éviter de favoriser les documents les plus longs, les vecteurs $[d]$ sont généralement normalisés en $[\underline{d}]$:

$$\underline{d_{ik}} = \frac{d_{ik}}{\sqrt{\sum_{k=1}^{|V|} d_{ik}^2}} \tag{2.3}$$

Cette dernière formule semble cette fois privilégier les documents les plus courts. A. Singhal [SBM96] suggère de multiplier ce poids par un terme correcteur supplémentaire fixé par validation croisée. Nous n'avons pas utilisé cette méthode dans nos travaux. Les formules suivantes utilisent toujours la pondération normalisée. Par simplification, nous noterons par la suite d_{ik} la valeur normalisée.

Pour la mesure de similarité entre documents, le modèle vectoriel préconise généralement la similarité dite du cosinus correspondant au cosinus de l'angle entre deux vecteurs, qui est égal au produit scalaire divisé par la norme des deux vecteurs. Comme les vecteurs sont déjà normalisés, le cosinus est égal au produit scalaire et directement relié à la distance euclidienne.

$$Cos(\mathbf{d_i}, \mathbf{d_j}) = \frac{\sum_{t_k} d_{ik} \cdot d_{jk}}{||\mathbf{d_i}||^2 ||\mathbf{d_j}||^2} = \sum_{t_k} d_{ik} \cdot d_{jk} = \langle \mathbf{d_i}, \mathbf{d_j} \rangle \tag{2.4}$$

$$\begin{aligned} ||\mathbf{d_i} - \mathbf{d_j}||_2^2 &= \sum_k (d_{ik} - d_{jk})^2 \\ &= \sum_k d_{ik}^2 + \sum_k d_{jk}^2 - 2 * \sum_k d_{ik} d_{jk} \\ &= 2 - 2\langle \mathbf{d_i}, \mathbf{d_j} \rangle \end{aligned} \tag{2.5}$$

Jones et Furnas [JF87] ont effectué une analyse géométrique des différentes mesures de similarité existantes (cosinus, produit scalaire sur vecteurs non normalisés, produit scalaire avec vecteurs normalisés suivant la norme L_1 et d'autres mesures moins utilisées). Ils concluent que le cosinus semble être le choix le plus judicieux car le moins sensible aux cas particuliers qui génèrent des comportements aberrants avec les autres mesures.

Le modèle probabiliste

Dans le modèle probabiliste, on considère que les documents sont générés par tirage aléatoire des différents termes qui les composent. Le *modèle de génération* décrit le processus sous-jacent qui génère les termes puis les documents. Ce processus n'est pas accessible aux observateurs, seul son résultat, c'est-à-dire la liste des documents du corpus qui ont été (supposément) générés par ce processus l'est. Les paramètres du processus, c'est-à-dire les valeurs de probabilités de chaque tirage, sont estimés à partir des occurrences trouvées sur les documents du corpus. Cela revient en général à estimer $P(t_k|c_j)$ la probabilité d'apparition du terme t_k sachant que le document appartient à la classe c_j. D. Lewis [Lew98] et K.S. Jones [JWR00] présentent le modèle probabiliste dans le cadre de la RI. Comme la représentation des documents est directement dépendante du modèle de génération utilisé, nous en présentons les différentes variantes dans la section 2.3.2.

2.3 Classifieurs issus de la RI

2.3.1 Rocchio

J. Rocchio a proposé un algorithme [Roc71] qui permet d'introduire une boucle de rétroaction avec intervention de l'utilisateur (*feedback*) pendant une recherche d'informations. Le principe de l'interaction est le suivant : l'utilisateur fournit une requête et reçoit une première liste de documents. Il peut alors sélectionner quelques documents en indiquant s'ils sont pertinents ou non pertinents. Ceux-ci sont utilisés par le système pour affiner la requête initiale (des termes présents dans les nouveaux documents sont ajoutés et les poids initiaux sont modifiés en fonction des occurrences des termes dans les nouveaux documents) et fournir une nouvelle liste de documents que l'on espère plus focalisée sur la recherche de l'utilisateur. La technique de Rocchio a été adaptée à la CT. Les informations concernant la requête initiale (qui n'existe pas en catégorisation classique) sont supprimées. L'algorithme calcule un profil prototypique $\mathbf{p_j}$ (c'est-à-dire un vecteur de l'espace de représentation des documents) pour chaque classe c_j qui correspond au barycentre des documents positifs et négatifs (avec un coefficient négatif pour les documents qui n'appartiennent pas à c_j) :

$$p_{jk} = \max \left\{ \begin{array}{l} \frac{t}{|c_j|} \sum_{d_i \in c_j} d_{ik} - \frac{1-t}{|\overline{c_j}|} \sum_{d_i \notin c_j} d_{ik} \\ 0 \end{array} \right. \tag{2.6}$$

où t est un paramètre libre compris entre 0 et 1. Les valeurs les plus utilisées sont 0.8 (les éléments positifs ont quatre fois plus d'importance que les éléments négatifs) et 1 (les éléments négatifs ne sont pas pris en compte du tout). L'utilisation des éléments négatifs dans le calcul du profil permet de mettre en avant non pas les mots les plus

fréquents mais les mots les plus discriminants pour la classification. Par exemple, si un terme apparaît aussi souvent dans un document de la classe que dans un document quelconque du corpus, le terme n'est pas utile pour déterminer l'appartenance à cette classe, son poids dans le profil doit donc être faible. L'affaiblissement du poids des termes fréquents est déjà en partie réalisé par l'IDF mais l'inclure dans le classifieur permet de le rendre plus spécifique à la tâche de classification. Le vecteur $\mathbf{p_j}$ est ensuite normalisé de la même façon que les documents de façon à ne pas favoriser une catégorie particulière lors de la classification.

$$\underline{p_{jk}} = \frac{p_{jk}}{\sqrt{\sum_k p_{jk}^2}} \tag{2.7}$$

Dans le cas où $t = 1$, la norme du vecteur p_j est égale à la similarité moyenne des paires de documents de la classe.

$$\begin{aligned} ||p_j|| &= \sqrt{\sum_k p_{jk}^2} \\ &= \sqrt{\sum_k \left(\frac{1}{|c_j|}\sum_{d_i \in c_j} d_{ik}\right)^2} \\ &= \sqrt{\frac{1}{|c_j|^2}\sum_k \left(\sum_{d_i, d_{i'} \in c_j} d_{ik} d_{i'k}\right)} \\ &= \sqrt{\frac{1}{|c_j|^2}\sum_{d_i, d_{i'} \in c_j} d_i \cdot d_{i'}} \\ &= \sqrt{\frac{1}{|c_j|^2}\sum_{d_i, d_{i'} \in c_j} sim(d_i, d_{i'})} \end{aligned} \tag{2.8}$$

La norme représente donc la densité (ou compacité) des classes. Plus la norme est grande et plus les documents sont proches les uns des autres. La norme est maximale pour une classe qui contient un seul document (car elle est alors égale à la similarité du document avec lui-même c'est-à-dire 1). En raison des termes diagonaux égaux à 1, qui ont un poids de $1/n$ sur la somme totale, plus le nombre de documents augmente, plus la norme diminue. Cette normalisation des profils permet donc de ne pas favoriser les classes les plus petites (ce qui serait contre-intuitif). E. Han a effectué une analyse plus précise de l'interprétation de la norme des prototypes dans [HK00].

La classification de nouveaux documents est effectuée en calculant la similarité entre ce document et les profils prototypiques et en affectant le document à la classe du prototype le plus proche.

$$f(d_i) = \text{argmax}_{c_j \in \mathcal{C}} \left(sim(d_i, c_j)\right) = \text{argmax}_{c_j \in \mathcal{C}} \left(\sum_{t_k} d_{ik} p_{jk}\right) \tag{2.9}$$

Interprétation géométrique

L'interprétation géométrique du modèle est très simple. Les documents sont représentés par un vecteur dans un espace de très grande dimension (de dimension la taille du

vocabulaire). La mesure de similarité entre deux documents correspond au cosinus de l'angle entre les deux vecteurs. Rocchio s'exprime facilement dans cette interprétation (voir figure 2.3.1). Les profils sont des vecteurs comme les documents. Dans le cas où $t = 1$, le profil $\mathbf{p_j}$ est alors le barycentre des exemples de la classe. Lorsque $t \neq 1$, les moyennes des exemples positifs et négatifs sont calculées et le profil est le barycentre de ces deux moyennes. Comme les exemples négatifs ont un coefficient négatif, ils ont tendance à écarter le profil du centre de l'espace de façon à rendre les différentes catégories plus distinctes.

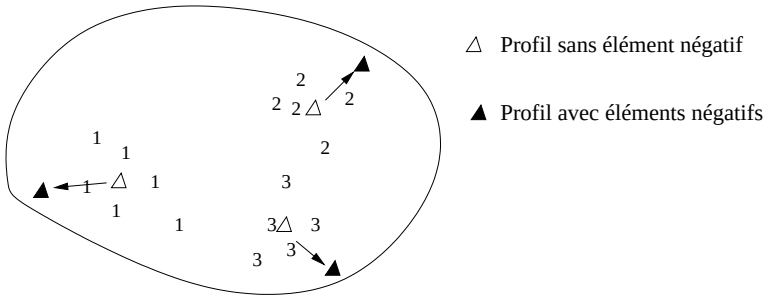


FIG. 2.1 – Interprétation géométrique de Rocchio.

Les améliorations de Rocchio

Si les algorithmes d'AA sont souvent plus performants que Rocchio, ce dernier est encore utilisé pour sa simplicité de mise en oeuvre et sa rapidité d'exécution. De multiples travaux ont été réalisés dans son prolongement pour l'améliorer. C. Buckley et G. Salton [BS95] utilisent la formule de Rocchio pour initialiser les poids puis les modifient itérativement à la manière d'un Perceptron en fonction des résultats sur un corpus de validation : c'est le *Dynamic Feedback Optimization*. Géométriquement, cela revient à optimiser le positionnement des prototypes en fonction des erreurs effectuées (se rapprocher des documents qui ont été mal reconnus ou s'éloigner des documents qui ont été injustement reconnus). Leurs performances en sont améliorées de 10 à 15 %. G. Karypis a également beaucoup étudié Rocchio et les algorithmes fondés sur des centroïdes [HK00] et a proposé un algorithme proche de celui de C. Buckley et G. Salton pour ajuster les poids des prototypes [SK00]. A. Singhal [SMB97] a affiné la formule de Rocchio en prenant en compte non pas tous les exemples négatifs mais uniquement ceux qui sont les plus proches du barycentre des exemples positifs (les p exemples les plus proches, ou les exemples dont la distance au barycentre est inférieure à une certaine valeur) : c'est le *Query Zoning*. L'idée sous-jacente est que les documents très éloignés du domaine de la classe ont un vocabulaire complètement différent et qu'il est inutile de les prendre en compte comme éléments négatifs puisque leurs mots n'apparaissent

de toute façon jamais dans les documents que l'on cherche à classer. Les seuls éléments négatifs à prendre en compte sont les documents dont le thème est très proche de celui recherché (voir la figure 2.2 pour une interprétation géométrique). Avec le Query Zoning, les auteurs obtiennent une amélioration comprise entre 5 et 10%. En notant $QZ(c_j)$ les "voisins" négatifs de la classe c_j, le calcul du poids s'exprime par :

$$R_{jk} = \max \begin{cases} \frac{t}{|c_j|} \sum_{d_i \in c_j} d_{ik} - \frac{1-t}{|QZ(c_j)|} \sum_{d_i \in QZ(c_j)} d_{ik} \\ 0 \end{cases} \quad (2.10)$$

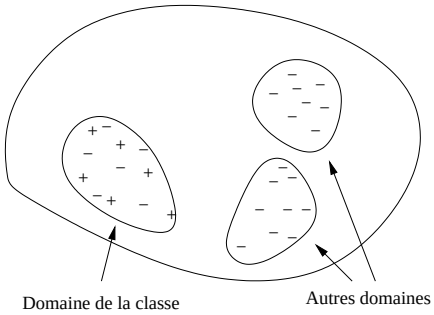


FIG. 2.2 – Interprétation géométrique du *Query Zoning*. Seuls les éléments négatifs du domaine doivent être pris en compte.

Schapire, Singer et Singhal ont intégré ces diverses techniques dans [SSS98] pour obtenir un classifieur comparable à ceux de l'état de l'art. Enfin, Moschitti [Mos03] améliore également les performances de Rocchio de 5 à 10% en optimisant la valeur du paramètre t indépendamment pour chaque catégorie. Cette optimisation est réalisée en effectuant une recherche exhaustive avec un corpus de validation.

2.3.2 Naïve Bayes

Si le modèle vectoriel et son principal algorithme Rocchio fonctionnent bien, leur fondement est entièrement empirique. Ils sont le résultat de nombreuses années de test. Le modèle probabiliste, au contraire, s'appuie sur une base théorique précise [JWR00]. Naïve Bayes est le représentant le plus populaire des classifieurs probabilistes. Le principe qui régit et donne son nom à l'algorithme est très simple. Il indique simplement que les différents attributs (dans le cas du texte, les différents termes présents dans le document) sont considérés comme indépendants. C'est la même hypothèse que pour le modèle vectoriel mais cette fois exprimée explicitement dans le cadre de la théorie probabiliste. Cet algorithme dont le modèle d'apprentissage est très général est utilisé dans de nombreux autres domaines que le texte.

La classification d'un exemple (un document dans le cas du texte) s'obtient par estimation de $p(c_j|d_i)$ la probabilité connaissant l'exemple d_i que celui-ci fasse partie de la

classe c_j. Le choix optimal (pour minimiser le taux d'erreur) est de mettre l'exemple dans la classe qui a la plus forte probabilité a posteriori. Comme cette probabilité n'est pas connue, il faut l'estimer à partir des données contenues dans le corpus d'apprentissage. La formule de Bayes permet "d'inverser" la probabilité conditionnelle :

$$P(c_j|d_i) = \frac{P(c_j) * P(d_i|c_j)}{P(d_i)} \tag{2.11}$$

Comme le but est de discriminer les différentes classes (il suffit d'ordonner les $P(c_j|d_i)$ pour toutes les classes ; il est inutile d'obtenir la valeur exacte), on peut alors supprimer le terme $P(d_i)$ qui est le même pour toutes les classes. $P(c_j)$ est la probabilité a priori qui est le plus couramment estimée par le pourcentage d'exemples appartenant à la classe c_j dans le corpus d'apprentissage.

$$\hat{P}(c_j) = \frac{N(c_j)}{N} \tag{2.12}$$

Le calcul de $P(d_i|c_j)$ dépend du modèle de génération des exemples. Dans le cas du texte, les plus populaires sont le modèle multivarié de Bernoulli et le modèle multinomial [Lew98, MN98].

Le modèle multivarié de Bernoulli

Dans le modèle le plus simple, un document est un vecteur binaire de la taille du vocabulaire. Il est généré par tirage aléatoire : chaque terme du vocabulaire peut être présent ou absent avec une certaine probabilité. Seule la présence/absence des termes est utilisée. Leur nombre d'occurrences dans le document n'a pas d'incidence. Le document se représente comme le résultat du tirage de V variables aléatoires indépendantes : $t_1, ..., t_V$. Pour rendre les formules exploitables, on fait de plus l'hypothèse d'indépendance des termes (hypothèse *Naïve* Bayes). $P(d_i|c_j)$ se simplifie alors en un produit de probabilités d'apparitions de chaque terme :

$$P(d_i|c_j) = \prod_{t_k \in d_i} P(t_k|c_j) \tag{2.13}$$

Il est possible d'estimer $P(t_k|c_j)$ à partir des exemples du corpus d'apprentissage. On utilise généralement l'estimateur du maximum de vraisemblance avec un lissage de Laplace (Laplace smoothing) pour éviter les probabilités nulles. Comme les probabilités de chaque terme sont multipliées, il suffit en effet qu'un seul terme dans un document à classer ne soit présent dans aucun document d'une classe (ce qui arrive souvent dans le domaine du texte où beaucoup de termes apparaissent très rarement) pour que la probabilité que le document appartienne à cette classe soit nulle. C'est pourquoi le lissage effectué pendant l'estimation est indispensable. On parvient alors, dans le cas du lissage de Laplace, à :

$$\hat{P}(t_k|c_j) = \frac{1 + \sum_{d_i \in c_j} 1_{ik}}{2 + |c_j|} \tag{2.14}$$

avec $1_{ik} = 1$ si $t_k \in d_i$, 0 sinon.

Le modèle multinomial

Dans le modèle précédent, il n'est pas possible d'utiliser les fréquences des termes dans les documents. Pour prendre en compte cette information supplémentaire, un autre modèle plus complexe a été proposé. Un document est une séquence de mots, chacun étant tiré aléatoirement parmi l'ensemble des mots du vocabulaire. Un document est donc généré par une distribution multinomiale des mots avec autant de tirages que de mots dans le document. Il est ainsi possible de prendre en compte la longueur des documents bien que cela soit rarement fait en pratique. L'hypothèse d'indépendance des termes reste nécessaire. $P(d_i|c_j)$ se réécrit en :

$$P(d_i|c_j) = P(|d_i|)|d_i|! \prod_{t_k \in d_i} \frac{P(t_k|c_j)^{Occ(i,k)}}{Occ(i,k)!} \tag{2.15}$$

Les probabilités sont une nouvelle fois estimées à partir des occurrences des exemples du corpus avec l'estimateur du maximum de vraisemblance et du lissage de Laplace :

$$\hat{P}(t_k|c_j) = \frac{1 + \sum_{d_i \in c_j} Occ(k,i)}{|V| + \sum_{k=1}^{V} \sum_{d_i \in c_j} Occ(k,i)} \tag{2.16}$$

Quant à la classification, l'estimation $\hat{P}(d_i|c_j)$ est calculée à partir des formules 2.13 et 2.15 suivant le modèle utilisé en remplaçant les probabilités $P(t_k|c_j)$ par leur estimateur $\hat{P}(t_k|c_j)$ et la classe c_j ayant la probabilité $\hat{P}(c_j|d_i)$ la plus élevée est choisie.

Le fonctionnement de Naïve Bayes est relativement similaire à celui de Rocchio. Chaque classe est décrite par un profil qui gère un coefficient par terme (p_{jk} pour Rocchio, $P(t_k|c_j)$ pour Naïve Bayes). Tous ces coefficients sont ensuite regroupés pour former une valeur de pertinence (un degré de similarité pour Rocchio, une probabilité pour Naïve Bayes). Il manque néanmoins à Naïve Bayes une meilleure prise en compte de la taille des documents et de l'importance logarithmique des fréquences des termes. C'est pourquoi, Naïve Bayes obtient en général des performances légèrement inférieures à Rocchio [Yan99].

Autres extensions du modèle

L'utilisation d'une hiérarchie dans les catégories (voir section 2.5) permet d'estimer les probabilités avec plus de précision par un lissage plus approprié que celui de Laplace [MRMN98, MN99]. Les probabilités $P(t_k|c_j)$ sont estimées à partir des occurrences

du terme t_k dans la classe c_j et des occurrences du terme t_k dans les classes parentes de c_j (qui contiennent plus de documents et peuvent donc calculer des estimations plus précises mais moins spécifiques). C'est la technique du *shrinkage*. [TCPH01, GGPC02] proposent un autre modèle génératif de documents qui tient compte de cette hiérarchie. D'autres travaux tentent de modifier le modèle pour éviter l'hypothèse d'indépendance des termes par les techniques de Maximum d'Entropie [NLM99] (le principe est ici d'utiliser un modèle exponentiel et d'estimer les paramètres en conservant des probabilités uniformes lorsqu'il n'y a pas d'information sur un attribut) ou de réseaux bayésiens [KS97] (les termes sont les sommets d'un graphe où chaque lien indique une dépendance entre deux termes). Les techniques mises en oeuvre dans les réseaux bayésiens permettent en général d'estimer les probabilités entre termes mais il est bien plus difficile de déterminer automatiquement la forme du graphe. Enfin, Joachims [Joa97] a analysé le modèle vectoriel et l'algorithme Rocchio sous un angle probabiliste. Il montre qu'en modifiant légèrement le facteur IDF, la normalisation des documents et en prenant le facteur TF le plus simple, Rocchio est identique à Naïve Bayes.

Malgré ces divers travaux, et contrairement au cas de Rocchio où les dernières avancées ont significativement augmenté les performances, l'algorithme de base n'a pas été amélioré de façon probante et il est désormais très en deçà des classifieurs les plus performants. Mais grâce à son efficacité en terme de vitesse d'exécution, il est actuellement le classifieur le plus utilisé dans les logiciels commerciaux de CT (en particulier pour le problème du filtrage du Spam, voir section 3.2).

2.4 Classifieurs issus de l'AA

L'Apprentissage Automatique (Machine Learning) est un vieux domaine initié dans les années 50 sous le nom de Reconnaissance des Formes. Une synthèse de ce domaine très vaste est disponible dans le livre "Machine Learning" de T. Mitchell [Mit97]. Nous nous contentons dans cette section de discuter des travaux d'AA qui concernent la CT. Une grande part des algorithmes d'AA ont été adaptés à la CT avec plus ou moins de succès. Les approches symboliques ont été utilisées dès le milieu des années 90, mais les algorithmes qui reposent sur des attributs binaires sont souvent incapables d'utiliser le nombre d'occurrences des termes, information pourtant très importante. Ces algorithmes tendent à disparaître peu à peu. Les classifieurs à base de règles de décision [Coh96, CS96], qui ont un fonctionnement proche de celui d'un moteur par mots-clés, ont peu à peu été remplacés par des arbres de décision [WAD+99, DPH98, LJ98].

Quelques travaux présentent l'utilisation de réseaux de neurones [Oar94] dans le cadre de l'apprentissage par les erreurs [DKR97] (les classifieurs ne sont mis à jour que lorsque des exemples sont mal classés), combinés à diverses méthodes de représentation

des textes [WPW95], à une hiérarchie de classes [RS02], ou dans un environnement dynamique [Str00].

Les algorithmes de classification linéaire sont très nombreux et beaucoup employés en CT. Leur point commun est de construire des surfaces de séparation entre classes qui sont linéaires, donc sous la forme d'un hyperplan. La méthode des moindres carrés [YC94], les méthodes à base de gradient (Widrow-Hoff, Exponential-Gradient ou Perceptron) [LSCP96] ou encore la régression logistique (la probabilité d'appartenance à une classe est modélisée par une combinaison linéaire des termes associé à une fonction logit : $\log(\frac{x}{1-x})$) [ZO01, ZY03, SHP95] ont été utilisées avec succès. Plusieurs études récentes semblent montrer que dans le cas du texte, les séparateurs linéaires sont suffisants pour discriminer les différentes classes [ZO01].

Les travaux semblent se concentrer aujourd'hui principalement autour de trois algorithmes : les k plus proches voisins, les Machines à Vecteurs Support (SVM) et le Boosting que nous détaillons ci-dessous.

2.4.1 K plus proches voisins (k-PPV)

L'algorithme des k plus proches voisins est un des plus vieux algorithmes de la reconnaissance des formes [CH67]. Il a été employé avec succès dans de nombreux domaines et a engendré toute une famille de classifieurs connus sous le nom de *classifieurs paresseux* (lazy learners) [Aha97]. Dans ces systèmes, le seul traitement effectué au cours de la phase d'apprentissage est le stockage des exemples sous une forme optimale de façon à pouvoir les extraire ensuite rapidement. Tous les calculs sont reportés à la phase de classification (d'où le terme de paresseux).

Dans les k-PPV, pour chaque nouvel exemple d à catégoriser, le classifieur calcule la similarité du document avec l'ensemble des autres exemples du corpus d'apprentissage. Il sélectionne ensuite les k documents les plus proches de d. La catégorie de d est attribuée par le vote (pondéré ou non par leur similarité) de ces k documents, donnant lieu aux formules 2.17 ou 2.18.

$$f(d) = \text{argmax}_{c_j \in \mathcal{C}}(\sum_{i=1}^{k} C(d_i, c_j)) \tag{2.17}$$

$$f(d) = \text{argmax}_{c_j \in \mathcal{C}}(\sum_{i=1}^{k} sim(d, d_i) * C(d_i, c_j)) \tag{2.18}$$

avec $C(d_i, c_j) = 1$ si d_i est de classe c_j, 0 sinon.

Pour que cet algorithme soit efficace, il faut une bonne mesure de similarité entre les documents, notamment afin que les attributs non discriminants ne soient pas pris en compte et que ceux qui sont discrimants le soient. Le modèle vectoriel a l'avantage de fournir une représentation dans laquelle les termes peu informatifs ont un poids faible.

La similarité en cosinus présentée dans la section 2.2.3 convient donc parfaitement et les k-PPV sont bien adaptés à ce domaine.

La phase d'apprentissage est souvent effectuée hors-ligne avant la mise en exploitation du logiciel. Il est donc acceptable de laisser l'algorithme tourner pendant un temps important. En revanche, pendant la phase de classification, les calculs doivent être très rapides (pour traiter un grand nombre de documents) voire quasiment instantanés si un utilisateur attend les résultats. L'algorithme des k-PPV a l'inconvénient de déporter tous les calculs qui pourraient être fait pendant la phase d'apprentissage à la phase d'exploitation Ce problème est d'autant plus important dans le cas du texte que les documents sont dans un espace de très grande dimension (le calcul de la similarité prend dans ce cas un temps non négligeable). La complexité des techniques d'accélération de recherches par structuration de l'espace (comme par exemple les *k-d tree* [FBF77]) varie exponentiellement avec la dimension de l'espace. Ils sont donc peu efficaces dans le cas du texte. En revanche, comme chaque document contient peu de termes, il est possible d'utiliser la méthode des index inversés (au lieu de stocker pour chaque document la liste des termes présents, on conserve pour chaque terme la liste des documents qui le contiennent), mais cela reste parfois insuffisant.

Dans les k-PPV, il n'existe pas de solution efficace pour choisir une bonne valeur pour le paramètre k. Ce choix relève d'un compromis. Si k est trop petit, le nombre d'exemples qui prennent part à la décision est faible et les exemples bruités peuvent alors jouer un rôle néfaste important. Si k est trop grand, l'hypothèse de localité n'est plus respectée car des exemples très éloignés du document sont sélectionnés pour participer au vote. Dans le domaine textuel, la valeur optimale pour k dépend du corpus et de l'application. D'après les travaux réalisés jusqu'à présent, la meilleure classification est obtenue avec une valeur de k comprise entre 10 et 50.

Les k-PPV ont été utilisés en CT pour la première fois par B. Masand [MLW92]. Y. Yang a montré que l'algorithme est parmi les meilleurs actuellement [YC94, Yan94, Yan99]. Ils sont très souvent utilisés comme élément de comparaison [Joa98, Coo99]. Quelques auteurs ont cherché à l'améliorer : L. Baoli cherche à optimiser la valeur de k pour chaque catégorie [BSQ03], W. Lam regroupe les documents dans des sous-classes [Lam98], E.-H. Han modifie la mesure de similarité en attribuant un poids à chaque terme par validation croisée [HKK01].

2.4.2 Machines à Vecteurs Support (SVM)

Le principe de SVM a été proposé par Vapnik à partir de la théorie du risque empirique et de la dimension de Vapnik-Chervonenkis (la VC-dimension correspond à une mesure de la difficulté d'un problème) [Vap95]. L'algorithme repose sur une interprétation géométrique simple [Bur98] : il s'agit de trouver la surface séparant les deux classes

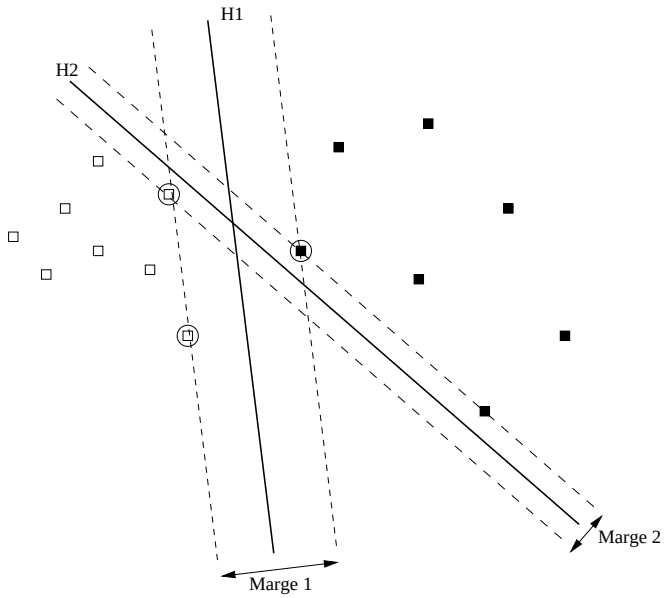


FIG. 2.3 – Exemples d'hyperplans séparateurs en dimension deux. Les vecteurs de support sont encerclés.

qui maximise la *marge*. La marge se définit comme la plus petite distance entre les exemples de chaque classe et la surface séparatrice S :

$$marge(S) = \sum_{c_j \in \mathcal{C}} min_{x_i \in c_j}(d(x_i, S)) \tag{2.19}$$

Dans l'exemple de la figure 2.3, les exemples des deux classes peuvent être séparés par un hyperplan, le problème est dit *linéairement séparable*. Les deux hyperplans H_1 et H_2 sont tous les deux des séparateurs acceptables, mais l'hyperplan H_1 a une plus grande marge et sera donc préféré. Pour calculer l'hyperplan optimal et donc la marge, seuls les exemples les plus proches de la zone-frontière sont mis à contribution. L'apprentissage consiste à déterminer ces exemples appelés vecteurs de support. Tous les autres peuvent être écartés et n'interviennent plus dans les calculs.

Le problème se traduit mathématiquement en un problème d'optimisation quadratique : trouver l'hyperplan $(\mathbf{w}, b)$ (b est la distance à l'origine de l'hyperplan) qui minimise la norme de w sous les contraintes :

$$\forall d_i,\ c_i(\mathbf{w}.d_i - b) \leq 1 \tag{2.20}$$

avec d_i le $i^{\text{ème}}$ document, de classe c_i ($+1$ ou -1). Si les exemples ne sont pas linéairement séparables, on peut les plonger conceptuellement dans un espace de dimension plus grande (la dimension peut même être infinie) par une fonction de transformation appe-

lée *noyau* (kernel). Dans cet espace, les exemples seront plus facilement séparables. Une propriété de l'algorithme est qu'il ne requiert pas les coordonnées de chaque exemple mais seulement les produits scalaires de chaque couple d'exemples, qui restent calculables une fois les exemples plongés dans un nouvel espace même de dimension infini. Si cela ne suffit pas pour rendre les exemples séparables, il est possible d'ajouter encore un terme correctif qui autorise un nombre limité d'exemples à être mal classés. Pendant l'apprentissage, on cherchera à rendre ce terme le plus petit possible. Un paramètre de l'algorithme permet de donner plus ou moins d'importance à ce terme correctif. Dans sa formulation initiale, SVM ne peut gérer que des problèmes bi-classes (des extensions commencent à apparaître pour faire du SVM multiclasse [HPRZ02, WW99, HL02]). La méthode la plus commune pour résoudre un problème multiclasse reste de le transformer préalablement en plusieurs sous-problèmes biclasse en utilisant les techniques présentées dans la section 2.1.

Cet algorithme est particulièrement bien adapté à la CT car il est capable de gérer des vecteurs de grande dimension et de sélectionner les attributs pertinents sans sur-apprentissage. Dans la pratique en CT, les catégories sont quasiment toujours linéairement séparables, il n'est donc pas nécessaire d'employer les méthodes avec des noyaux sophistiqués qui alourdissent inutilement les calculs. SVM a été introduit en CT pour la première fois par Joachims [Joa98] qui a notamment travaillé à rendre SVM compatible avec les données typiques de la CT (grande dimension, matrices très creuses). Depuis, l'algorithme a été très souvent réutilisé par exemple pour la catégorisation de dépêches [Coo99] ou la détection de courriers électroniques non sollicités [DWV99, KA01]. E. Leopold et J. Kindermann ont cherché quelle était la meilleure représentation des textes pour SVM [LK02]. Quelques travaux cherchent à améliorer encore l'algorithme en l'adaptant à un contexte dynamique [Kwo98], en modifiant la phase d'optimisation pour la rendre plus rapide [DPH98] ou en adaptant l'algorithme pour le rendre à même de traiter des corpus de très grande taille [Joa99].

2.4.3 Boosting

Le *boosting* [FS97] est une instance des classifieurs par comités [LC96, Bre94] dont le principe est d'aggréger les résultats de plusieurs classifieurs pour obtenir un résultat plus robuste. Les différents classifieurs peuvent correspondre à différents algorithmes [LW94] ou au même algorithme utilisé avec différents sous-échantillons du corpus d'apprentissage [DE95]. Dans le boosting (voir algorithme 1), les différents classifieurs sont appris séquentiellement. À chaque étape, le corpus d'origine est échantillonné suivant une distribution qui favorise le tirage des exemples mal classés par le classifieur construit à l'étape précédente. Les classifieurs se focalisent ainsi de plus en plus sur les éléments difficiles à catégoriser. Lors de la phase de test, tous les classifieurs sont utilisés et un

Algorithme 1 AdaBoost

Initialiser la distribution D_1 par $D_1(i) = 1/N$ pour tout i
pour tout t de 1 à T **faire**
Apprendre un classifieur f_t à partir de la distribution D_t et de l'algorithme de classification.
Calculer l'erreur $\epsilon_t = \sum_{i:f_t(i)\neq y_i} D_t(d_i)$
Soit $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
Mettre à jour la distribution : $D_{t+1}(d_i) = \frac{D_t(d_i)}{Z_t} * \begin{cases} e^{-\alpha_t} \text{ si } f_t(d_i) = y_i \\ e^{\alpha_t} \text{ si } f_t(d_i) \neq y_i \end{cases}$
avec Z_t un facteur de normalisation de façon à ce que D_{t+1} soit bien une distribution de probabilité.
fin du pour
Produire le classifieur final : $f(d) = sign\left(\sum_{t=1}^{T} \alpha_t f_t(d)\right)$

vote pondéré par les taux de performance de chaque système est effectué. Le boosting est souvent utilisé avec un algorithme peu performant mais très rapide (surnommé Weak Learner) avec lequel on construit plusieurs centaines de classifieurs. L'algorithme garantit que le taux d'erreur sur le corpus d'apprentissage peut être rendu aussi petit que l'on veut en augmentant simplement le nombre de classifieurs construits. Les expériences menées montrent que, en pratique, les performances en généralisation sont également très bonnes (il n'y a pas de sur-apprentissage) [SFBL97, DC96].

Comme SVM, le boosting a été adapté à la CT avec succès [SSS98, lLS+00, KHZ00, SS00].

2.5 Classer dans une hiérarchie

Lorsque le nombre de classes est relativement important (au-delà de 10), cela signifie que le domaine de l'application est assez vaste. Il est alors fréquent de structurer les thèmes sous la forme d'une hiérarchie. Les catégories peuvent correspondre uniquement aux feuilles ou à tous les noeuds de l'arbre.

À partir de l'intuition qu'il est plus facile de discriminer les classes du plus général au plus précis, quelques travaux ont essayé de prendre en considération une telle hiérarchie pour améliorer la classification. L'idée la plus simple consiste à apprendre un classifieur par nœud, puis lors de la classification, descendre dans l'arbre en suivant le noeud proposé par chaque classifieur. Par exemple, on commence par discriminer le sport de l'économie puis, une fois que l'on a décidé que le document parlait de sport, on différencie le football du tennis. Avec ce principe, les résultats en terme de justesse des prédictions sont peu convaincants (pas de différence significative par rapport à un classifieur "plat"). En revanche, l'utilisation de la hiérarchie permet en général un gain en temps de traitement important (entre 2 et 3 fois plus rapide). Ce gain a été également

analysé à travers la théorie de la complexité dans [YZK03].

Ces résultats sont en accord avec l'étude théorique de Mitchell [Mit98] qui montre, dans un cadre probabiliste, l'équivalence entre classifieur hiérarchique et classifieur plat lorsque plusieurs conditions sont vérifiées. La première est le fait que la classification doit se faire dans toutes les branches au lieu de prendre seulement le chemin le plus probable. Cette hypothèse semble encore défavoriser les classifieurs hiérarchiques puisque dans l'utilisation classique la classification se fait par le chemin le plus probable, ce qui semble a priori moins performant que de visiter toutes les branches. Une deuxième condition nécessaire est de ne pas lisser les probabilités (voir section 2.3.2 à propos du lissage de Laplace).

Une utilisation alternative de ces hiérarchies est proposée par McCallum [MRMN98] pour améliorer le lissage et ainsi s'éloigner des conditions d'équivalence mais les effets restent limités. Une autre étude propose d'utiliser la hiérarchie pour faire une forte sélection sur les attributs à conserver [KS97, Mla98b]. Cette solution n'améliore pas les performances mais permet encore un gain en temps de calcul. Une troisième possibilité consiste à développer des classifieurs prenant directement en compte la hiérarchie [WZL99, GGPC02]. D'Alessio [DMKS00] a montré que la hiérarchie du corpus Reuters n'était pas optimale et qu'il était possible d'améliorer les performance en déplaçant des branches de l'arbre d'un noeud à un autre. Ces modèles sont encore au stade expérimental et peu de résultats réellement satisfaisants ont été obtenus.

2.6 Évaluation des classifieurs textuels

Dans le domaine de la RI et de l'AA, la validation des méthodes est généralement empirique. Il existe plusieurs grandes campagnes d'évaluation internationales, chacun se focalisant sur une tâche particulière (TREC [NDa], TDT [NDb], MUC [ARP], Senseval [AS] par exemple) dont le but est de comparer objectivement différents systèmes sur les mêmes données et avec les mêmes mesures de performance. Malgré ces campagnes, il reste très difficile d'affirmer la supériorité d'un algorithme sur un autre car les résultats sont très dépendants des corpus utilisés, de la tâche à évaluer, des mesures de performance et même des implémentations [Sal97].

2.6.1 Mesures de performance

Il y a deux critères importants dans l'évaluation d'un algorithme de catégorisation. L'*efficacité* est le critère technique qui mesure les temps de calcul et la taille mémoire nécessaire. La *justesse des prédictions* mesure si la catégorisation effectuée est correcte.

Pour mesurer l'efficacité, il est possible d'utiliser la théorie de la complexité. Cette dernière permet de partager les différents algorithmes en plusieurs grandes classes (po-

lynomiaux, exponentiels) mais les analyses théoriques des algorithmes sont souvent difficilement comparables. Il est aussi possible d'utiliser des mesures empiriques pour comparer les temps de calcul. Malheureusement, ces mesures dépendent fortement des conditions d'expérimentations et sont difficilement réutilisables. Par ailleurs, l'évolution de la puissance de calcul des ordinateurs rend ces résultats très rapidement obsolètes. Il existe très peu d'études probantes concernant cette question à l'exception d'un travail de Y. Yang [YZK03] et les résultats restent très qualitatifs.

En revanche, de très nombreux travaux cherchent à comparer les performances de différents algorithmes en terme de justesse des prédictions. Ces comparaisons sont essentiellement empiriques à partir de mesures effectuées sur des corpus de test. Il faut donc découper un corpus en deux parties, l'un servant à l'apprentissage et l'autre aux tests. Suivant le découpage effectué, les résultats peuvent être significativement différents. Ce problème a été largement étudié en statistique à travers les théories de l'échantillonnage. Plusieurs méthodes ont été proposées pour le résoudre [DK95, Die97, Sal97]. La plus connue est la technique de validation croisée. Le corpus est découpé en p parties de même taille (souvent 10). Le test se fait en p phases. À chaque étape, $p-1$ ensembles sont regroupés pour former le corpus d'apprentissage et le classifieur est testé sur les exemples du dernier ensemble. Le taux de performance est calculé comme la moyenne des taux sur chaque essai. Dans le "leave-one-out", les sous-ensembles sont constitués d'un seul exemple ; l'apprentissage se fait donc sur tout le corpus sauf un exemple, celui qui est testé.

Il est, de plus, nécessaire de faire les tests sur de nombreux corpus différents afin que les résultats soient généralisables sur d'autres corpus non testés. Il existe de nombreuses mesures pour rendre compte des résultats. Elles dépendent essentiellement du modèle de tâche pour laquelle le système de CT est utilisé. Toutes ces mesures utilisent les valeurs de la matrice de confusion (qui contient à la position (i, j) le nombre d'exemples de la classe c_j catégorisés comme c_i) ou la table de contingence pour une classe précise qui correspond à la matrice de confusion dans le cas binaire (voir le tableau 2.2).

		Classe correcte	
		Classe 1	Classe 2
Classe prédite	Classe 1	A Vrais positifs	B Faux positifs
	Classe 2	C Faux négatifs	D Vrais négatifs

TAB. 2.2 – Table de contingence (matrice de confusion dans le cas binaire).

Précision-rappel

La mesure la plus utilisée est le couple *précision-rappel* (precision et recall en anglais) développée en premier lieu pour la RI. Elle est employée lorsque le système de CT doit prendre une décision binaire pour chaque exemple : le document d_i appartient-il, oui ou non, à la catégorie c_j ? Les deux classes 1 et 2, représentées dans le tableau 2.2, n'ont donc pas le même rôle (la classe 1 correspond à la réponse oui, la classe 2 à la réponse non).

$$\text{Précision} : p = \frac{A}{A+B} \tag{2.21}$$

$$\text{Rappel} : r = \frac{A}{A+C} \tag{2.22}$$

La mesure F_β combine ces deux valeurs :

$$F_\beta = \frac{(\beta^2+1)pr}{\beta^2 p + r} \tag{2.23}$$

Lorsque $\beta > 1$, la précision joue un rôle plus important que le rappel et la mesure F_β favorise les classifieurs avec une bonne précision. Inversement lorsque $\beta < 1$, le rappel est favorisé. Lorsqu'il n'y a pas d'a priori, la valeur $\beta = 1$ est utilisée.

Dans la plupart des systèmes, la sortie de l'algorithme est, pour chaque classe, une valeur numérique qui indique la "confiance" que le classifieur accorde à son propre jugement. Lorsque la valeur dépasse un certain seuil, le document est classé comme positif, sinon il est classé négativement. En fonction du choix de la valeur du seuil, il est possible d'être plus ou moins sélectif : ranger beaucoup de documents dans la classe pour s'assurer d'obtenir tous les documents de cette catégorie, au risque de trouver des documents qui n'y appartiennent pas (rappel élevé et précision basse) ou au contraire restreindre la sélection pour n'avoir que des documents qui appartiennent effectivement à cette classe, quitte à en oublier quelques-uns (précision élevée et rappel bas). Pour pouvoir visualiser les performances en fonction des différents comportements souhaités, il est fréquent d'utiliser une courbe de précision-rappel qui est construite en mesurant la précision pour onze valeurs précises de rappel (0, 0.1, ..., 0.9, 1) (la valeur du seuil est modifiée à chaque fois pour obtenir la valeur de rappel souhaitée et on mesure la précision obtenue avec ce paramétrage). Le graphe présente les onze points reliés avec le rappel en abscisse et la précision en ordonnée. Une autre mesure qui agrège les résultats est le *break-even point*, valeur pour laquelle le rappel est égal à la précision. Les évaluations des conférences TREC pour la tâche de filtrage [Hul98], quant à elles, utilisent le principe des mesures *d'utilité*. Elles sont fondées sur les critères de *gain* et de *perte*. La performance est une combinaison linéaire des quatres valeurs de la table de contingence (pour TREC : $\alpha = 2$, $\beta = -1$, $\gamma = \delta = 0$).

$$U = \alpha A + \beta B + \gamma C + \delta D \tag{2.24}$$

Taux d'erreur et de performance

Lorsque le problème est multiclasse (un document doit être catégorisé dans une et une seule classe), les mesures de précision-rappel ne sont pas adaptées. En effet, lorsque les classes sont prises en compte simultanément, le rappel et la précision deviennent liés : une erreur qui fait baisser le rappel dans une classe fait aussi baisser la précision dans une autre classe. On utilise alors généralement soit le taux d'erreur, soit son opposé, le taux de bonne classification (accuracy en anglais) :

$$\text{Taux d'erreur : } e = \frac{B + C}{A + B + C + D} \tag{2.25}$$

$$\text{Taux de bonne classification : } perf = \frac{A + D}{A + B + C + D} \tag{2.26}$$

Ces deux mesures se généralisent pour plus de deux classes ($A + D$ est remplacé par la somme de tous les éléments diagonaux de la matrice de confusion, $B + C$ est remplacé par la somme de tous les autres éléments, c'est-à-dire le nombre d'exemples mal classés).

Micro et macro-moyenne

Toutes ces mesures, fondées sur l'utilisation de la table de contingence, sont définies pour une catégorie. Il existe deux méthodes pour agréger les résultats de chaque classe. Soit on construit une table de contingence globale ($A = \sum_{c_j} A_j$ et de même pour B, C et D) et on calcule les valeurs $p, r, e, perf$ à partir de cette matrice : c'est la *micro-moyenne*. Soit on calcule p_j, r_j, e_j et $perf_j$ pour chaque classe et on fait la moyenne de chaque valeur ($p = 1/|\mathcal{C}| \sum_{c_j} p_j$) : c'est la *macro-moyenne*. Si les classes ne sont pas équi-distribuées, les deux moyennes sont différentes : la micro-moyenne donne plus de poids aux classes de grande taille alors que la macro-moyenne donne plus de poids aux petites classes. Si la majorité des études utilisent la micro-moyenne, le choix de privilégier l'une ou l'autre de ces mesures dépend intrinsèquement de l'application dans laquelle sera utilisé le système.

2.6.2 Corpus standards

Afin de pouvoir comparer les performances obtenues par divers algorithmes, il est nécessaire de les tester sur les mêmes corpus. Quelques bases de textes ont donc émergé comme corpus de référence pour la CT. L'utilisation de ces corpus permet ainsi une comparaison plus aisée entre différentes techniques. Mais leur omniprésence dans les évaluations favorise les algorithmes optimisés sur les types de tâches proposés par ces

corpus et limite l'apparition d'algorithmes dédiés à des applications non standard. De plus, l'avantage d'utiliser des corpus standard (pouvoir comparer les performances entre deux études différentes) est affaibli par les variations importantes dans les protocoles d'expérimentation (les mesures de performance, le découpage apprentissage/test, les prétraitements utilisés). Y. Yang a mis cet aspect en évidence dans son étude qui synthétise les résultats obtenus sur le corpus Reuters [Yan99].

Les corpus les plus utilisés sont les suivants :

- Le corpus Reuters-22173 concerne la classification de dépêches de l'agence de presse Reuters. Malgré la quasi-unanimité des chercheurs à reconnaître sa mauvaise qualité (les catégories sont mal définies, leur répartition est trop déséquilibrée), les travaux continuent à utiliser ce corpus pour lequel on connaît les performances de très nombreux algorithmes. Reuters a récemment publié un nouveau corpus de plus grande taille et de meilleure qualité nommé Reuters2000[1] (les deux corpus sont comparés dans [CS02]).
- Le corpus *20 Newsgroups*[2], créé par Lang [Lan95], est constitué de 20 000 messages équirépartis dans 20 groupes de discussion Usenet. Ces 20 classes se placent naturellement dans une hiérarchie propre à Usenet. Elles permettent de faire des expériences avec des algorithmes qui prennent en compte une hiérarchie sur les classes.
- Le corpus AP contient des dépêches de presse. C'est un extrait du corpus des conférences TREC [NDa].
- Le corpus OHSUMED, créé par Hersh et al. [HBLH94] contient près de 350 000 titres et résumés d'articles de journaux scientifiques issus du domaine médical, extrait du corpus MEDLINE qui contient sept millions de référence. Les classes correspondent aux catégories du thésaurus MeSH (Medical Subject Headings).
- Le corpus *WebKB*[3], créé par Craven et al. [CDF+98], contient des pages web extraites de quatre universités américaines importantes. Les catégories correspondent au type des pages : étudiant, université, personnel, cours, projet, département et autre.
- Le corpus *Yahoo Science*, proposé pour la première fois par McCallum [MRMN98], contient les pages d'accueil de sites indexés dans l'annuaire Yahoo sous la rubrique Science. Les classes correspondent aux sous-catégories de Science dans cet annuaire.

Le tableau 2.3 présente une vue d'ensemble de l'utilisation de ces corpus à travers quelques publications scientifiques.

[1] `http://about.reuters.com/researchandstandards/corpus/`
[2] `http://www.ai.mit.edu/~jrennie/20Newsgroups`
[3] `http://www.cs.cmu.edu/~webkb`

Titre des publications	Reuters	20 Newsgroups	AP	OHSUMED	WebKB	Yahoo Science
A neural network approach to topic spotting de Wiener et al. [WPW95]	X					
Training algorithms for linear text classifiers de Lewis et al. [LSCP96]			X	X		
A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization de Joachims [Joa97]	X	X				
Inductive learning algorithms and representations for text categorization de Dumais et al. [DPH98]	X					
Automated text categorization using support vector machine de Kwok [Kwo98]	X					
Improving Text Classification by Shrinkage in a Hierarchy of Classes par McCallum et al. [MRMN98]		X				X
Distributional clustering of words for text classification de Baker et McCallum [BM98a]	X	X				
Error-correcting output coding for text classification de Berger [Ber99]		X		X		X
Using Maximum Entropy for Text Classification de Nigam et al. [NLM99]		X			X	
Maximizing text-mining performance de Weiss et al. [WAD+99]	X					
BoosTexter : A boosting system for text classification de Schapire et Singer [SS00]	X	X	X			
A Statistical Learning Model of Text Classification with Support Vector Machines de Joachims [Joa01]	X			X	X	
Text Categorization Based on Regularized Linear Classification Methods de Zhang et Oles [ZO01]	X	X			X	
Improving Text Classification with LSI Using Background Knowledge de Zelikovitz et Hirsh [ZH01]		X			X	
A New Family of Online Algorithms for Category Ranking de Crammer et Singer [CS02]	X	X				
A study on optimal parameter tuning for Rocchio text classifier par Moschitti [Mos03]	X			X		

TAB. 2.3 – Liste de corpus utilisés dans diverses publications.

2.7 Conclusion

La CT s'est établie au cours de la dernière décennie comme un domaine majeur de recherche pour les systèmes d'informations. Ce dynamisme est en partie dû à la demande importante des utilisateurs pour cette technologie. Elle devient de plus en plus indispensable dans de nombreuses situations où la quantité de documents textuels électroniques rend impossible tout traitement manuel.

Dans sa version "simple", c'est-à-dire avec des catégories thématiques qui correspondent au sujet principal du document, la tâche de classification et toutes ses variantes (filtrage, routage, classement) sont désormais bien maîtrisées. Les algorithmes existants pour numériser les textes et faire l'apprentissage automatique sont maintenant bien connus. Les protocoles de tests sont documentés puisque adaptés des modèles de la reconnaissance des formes et de grandes campagnes d'évaluation internationales ont été mises en place.

La CT a essentiellement progressé ces dix dernières années grâce à l'introduction des techniques héritées de l'apprentissage automatique qui ont amélioré très significativement les taux de bonne classification. L'algorithme SVM (Support Vector Machine) semble actuellement le plus performant des classifieurs. De plus, son paramétrage par défaut est approprié dans la majorité des situations ce qui permet d'éviter une fastidieuse étape de réglages.

Il reste nénamoins difficile de fournir des valeurs chiffrées sur les performances qu'un système de CT peut actuellement atteindre. La tâche est souvent subjective : lorsque deux experts humains doivent déterminer les catégories d'un ensemble de documents, il y a souvent désaccord sur plus de 5% des textes. Il est donc illusoire de rechercher une classification automatique parfaite. Actuellement, les systèmes de CT avec apprentissage automatique ont rattrapé les performances des systèmes manuels qui nécessitent de longs mois de développement à des experts humains pour fournir les règles de catégorisation. Les meilleurs algorithmes obtiennent sur les corpus standards (Reuters, Newsgroups, Oshumed) souvent plus de 90% de bonne classification.

Il reste néanmoins difficile d'extrapoler cette valeur sur d'autres corpus et applications. Les résultats sont extrêmement dépendants du type des documents et des catégories (en particulier de leur nombre). Il n'existe pas, à l'heure actuelle, d'analyse de la performance des algorithmes en fonction des spécificités des corpus. Ce manque de garantie sur un procédé intrinsèquement incertain est un faceur qui limite son déploiement à grande échelle.

CHAPITRE

3

De nouvelles applications pour la CT

Sommaire du chapitre

3.1 Introduction

NOUS AVONS passé en revue dans le chapitre précédent les diverses techniques utilisées dans le cadre de la Catégorisation de Textes (CT), qu'elles soient issues de la Recherche d'Informations (RI) ou de l'Apprentissage Automatique (AA).

Ces techniques ont surtout été employées dans un cadre classique de catégorisation thématique. Dans ce contexte, les classes correspondent alors au thème central des

documents. La principale application de la CT est le classement de documents dans une bibliothèque numérique. Plusieurs applications peuvent être dérivées de cette tâche générique. La plus connue est la catégorisation d'articles de journaux par rubrique (économie, politique, sport) [RK98, Coo99, BSH00]. Le corpus Reuters [Yan99] en est une illustration. Le classement de sites web dans des annuaires de type Yahoo est du même type. Dans ces applications, les catégories sont généralement structurées sous la forme d'une hiérarchie de thèmes qu'il est possible d'utiliser pour améliorer les prédictions (voir section 2.5). Une autre application concerne le classement d'articles scientifiques à partir du titre et du résumé. Cette tâche a été beaucoup étudiée à partir du corpus OSHUMED qui répertorie 350 000 articles du domaine médical [HBLH94] et du portail scientifique Cora [MNRS00]. Un dernier type d'application est le classement de documents spécialisés comme les brevets [Lar98, IFKM03].

Depuis quelques années, de nouvelles applications se mettent à utiliser la CT dans un cadre légèrement différent du modèle initial. Les catégories à assigner aux documents ne portent plus sur le contenu thématique ou lexical du texte mais sur d'autres propriétés, qui nécessitent souvent une compréhension plus poussée du texte.

La catégorisation non-thématique d'images (images d'intérieur ou d'extérieur par exemple) à partir des légendes [SH99], l'attribution d'une note à des dissertations d'étudiants [Lar98], le classement de courriers électroniques dans les dossiers de chaque utilisateur [Moc99, SK99b, Ren00] ou encore les solutions de dialogue par serveur vocal pour résoudre un problème ou router l'appel vers un opérateur compétent [CCC99, GAA+02] font partie de ces nouvelles applications. Dans ce dernier exemple, les techniques de CT doivent de plus être adaptées pour prendre en compte le fait que les textes correspondent à la sortie des modules de reconnaissance de la parole et qu'ils sont donc susceptibles de contenir des mots erronés.

Nous présentons dans ce chapitre quatre de ces nouvelles applications, qui sont celles que nous avons étudiées au cours de cette thèse : le filtrage de courriers électroniques non sollicités (pourriels ou spam en anglais), la détection des points de vue exprimés dans un document, la sélection d'une réponse à une question dans le cadre du Mail Center et le classement de dépêches suivant l'évènement générateur.

3.2 Le courrier électronique non sollicité

3.2.1 Présentation du problème

Les courriers électroniques non sollicités, communément appelés Spam[1] ou pourriels, correspondent aux emails, généralement à vocation commerciale, envoyés sans l'autorisation du destinataire alors que ce dernier ne souhaite pas les recevoir. Parce que l'envoi

[1] Le terme SPAM provient d'un sketch des Monty Python dans lequel une conversation est peu à peu dérangée par un groupe de vikings répétant inlassablement "spam, spam, spam, spam...".

d'un email ne coûte presque rien, ces courriers sont souvent envoyés à très grande échelle sans souci de cibler les utilisateurs intéressés. C'est cet envoi massif qui caractérise le Spam. Les adresses emails sont collectées à l'insu des utilisateurs par des robots qui parcourent le web et les groupes de discussions Usenet. Plusieurs études de Message Labs, Jupiter Research, Ferris Research ou encore BrightMail, indiquent que le phénomène du Spam croît de presque 400% par an et qu'il commence à avoir un impact non négligeable sur la productivité des salariés en entreprise ou le dimensionnement des réseaux nécessaire au transport des emails. Le Spam représenterait aujourd'hui près de 50% des messages reçus par les utilisateurs.

L'analyse des corpus de Spam [CNI02, OK02] montre que quelques thèmes suffisent à décrire l'ensemble de ces courriers (la figure 3.1 présente deux exemples typiques de Spam) :

- le courrier commercial (publicité) non sollicité,
- les messages offrant des opportunités financières avec de grandes récompenses pour très peu d'effort et de risque (par exemple les lettres-chaînes, les demandes de transfert de fond ou de refinancement),
- les produits de santé "miraculeux" (pour perdre du poids ou améliorer ses performances sexuelles),
- les messages à caractère illégal (mais qui ne sont pas forcément illégaux dans le pays d'envoi) : pornographie, jeux d'argent, produits interdits à la vente,
- la propagation de rumeurs (hoax) comme les mises en garde contre de prétendus virus.

Le Spam regroupe des emails de différentes langues, ce qui multiplie le nombre de thèmes apparents. En effet, comme les langues ne partagent pas leur dictionnaire, les profils lexicaux d'un même thème dans deux langues différentes sont complètements différents. Une autre caractéristique du Spam est l'évolution temporelle des thématiques : certaines sont constantes, alors que d'autres apparaissent ou disparaissent régulièrement.

Le filtrage du Spam était traditionnellement effectué en collectant les adresses des machines émettrices dans un système de "liste noire" (blacklist). Les systèmes plus sophistiqués intègrent également des règles basées sur le contenu des en-têtes ou des informations péri-textuelles du message. Ces méthodes sont néanmoins insuffisantes et l'ampleur actuelle du Spam nécessite la mise en place de nouvelles techniques. Comme les thèmes sont peu nombreux et relativement bien connus, il est possible d'analyser le contenu des emails pour y trouver un des thèmes du Spam. Ceci peut être effectué de façon rudimentaire à partir d'une liste de mots-clés ou avec des systèmes de CT.

Les premiers travaux à avoir utilisé la CT pour le filtrage du Spam remontent, selon nous, à M. Sahami en 1998 avec les réseaux bayésiens [SDHH98]. Depuis, de nombreuses techniques ont été essayées comme Naïve Bayes, k-PPV [AKCS00], SVM [DWV99], ou

Sujet : Order your Viagra and Weight-loss Here
VIAGRA WITHOUT A DOCTORS VISIT!!
CLICK HERE
*Other Top Medications also available!!
*We have Doctors on call around the country to view your information and quickly approve your order.
*Totally Discreet System allows you to order today and enjoy your medication tomorrow in most cases.
*Finally you can try the wonder drug Viagra that has swept the World without the embarrassment of having to visit your Doctor and explain your condition!!
TO ORDER CLICK HERE!

Sujet : Double your money in 30 days
Doubling your investment is more achievable than you think!
200% return could have been achieved in less than 30 days! Learn about our strategy trading in the Interbank Foreign Currency Exchange.
Get our Reports, Charts and Strategies on the US Dollar Vs Japanese Yen and Euro.
Example : A $10,000 Investment in the Euro Vs the dollar, "properly positioned", on 7/17/01 could possibly have returned $25,000 on 08/10/01.
Includes initial investment and excludes any commission and fees and markups. Must be 21 years of age. Risk Capital only. Past performance not indicative of future performance. For additional information : See risk disclosures on request form.

FIG. 3.1 – Exemples de Spams

Boosting avec des arbres de décision [CM01]. [GHMLPS00, KA01] s'interrogent plus particulièrement sur l'aspect dissymétrique de la tâche. En effet, l'erreur de classer un email comme Spam alors que c'est un courrier légitime est beaucoup plus grave que l'erreur inverse. Dans le premier cas, le mail légitime sera supprimé sans avoir été lu alors que dans le second cas, l'utilisateur devra simplement lire un Spam qui n'aura pas été filtré.

Les systèmes adaptatifs avec CT sont généralement très performants avec plus de 99% de bonne classification (voir par exemple `http://crm114.sourceforge.net/` et `http://www.nuclearelephant.com/projects/dspam/`) et ils commencent à être intégrés dans les applications réelles (voir [BGT03] pour une liste d'applications commercialisées de filtrage de Spams avec CT). Deux conférences sur le Spam ont eu lieu récemment au MIT en 2003 et 2004[2].

Ce qui nous intéresse tout particulièrement dans cette problématique est la multithématicité des catégories. En effet, par définition, ce n'est pas le contenu mais le côté non sollicité du courrier qui caractérise un Spam, ce qui n'est pas une définition thématique de la catégorie. Il est néanmoins possible de lister l'ensemble des thèmes qui

[2] `http://www.spamconference.org`

apparaissent dans les Spams. La catégorie peut alors se définir non pas par une mais par plusieurs thèmes. Le nombre de thèmes est, de plus, multiplié par le nombre de langues.

3.2.2 Expérimentations avec le projet *Spam Buster*

Pour traiter ce problème, nous avons réalisé le projet SpamBuster[3] qui consiste en un filtreur de Spams par CT. Ce service est accessible depuis 2001 à l'ensemble du personnel du Département Informatique et Réseaux de l'ENST. Une fois l'utilisateur inscrit, tous ses emails passent par le système de CT et un en-tête indiquant leur nature (Spam ou non) est ajouté au mail. Lorsque la classification est erronée, l'utilisateur est invité à renvoyer le mail erroné à une adresse spécifique, ce qui nous permet d'intégrer ce nouvel exemple au corpus d'apprentissage. Ce dernier est donc alimenté peu à peu par les erreurs de classification.

Il est clair que l'adaptation de l'algorithme aux nouveautés est un point crucial pour le problème du Spam. Les spammeurs faisant tout pour éviter de se faire reconnaître, ils adaptent régulièrement leurs techniques pour ne pas être filtré. Un système de filtrage doit donc évoluer également pour prendre en compte ces changements. La meilleure méthode est clairement d'obtenir régulièrement l'ensemble des Spams et des courriers légitimes reçus et de faire un apprentissage avec tous ces documents. Ceci est néanmoins trop intrusif pour les utilisateurs. Il faut donc trouver un compromis pour obtenir un système le plus réactif possible (ce qui nécessite un maximum de documents étiquetés) tout en limitant les interventions de l'utilisateur pour faire cet étiquetage. Nous avons choisi, dans le SpamBuster, de ne lui fournir que les documents mal classés. L'apprentissage par les erreurs semble une stratégie simple et efficace (certains algorithmes comme par exemple les réseaux de neurones fonctionnent par défaut de cette manière : seuls les documents mal classés sont utilisés pour modifier incrémentalement les poids du réseau). L'autre possiblité est d'utiliser les méthodes d'*apprentissage actif* dans lesquelles le système choisit les documents qui doivent être présentés à l'utilisateur pour que celui-ci donne la classe du document [TK00].

Pour avoir un corpus fixe lors de nos expériences, nous avons pris un instantané du corpus d'apprentissage. La base ainsi constituée contient 2193 courriers dont 733 Spams (660 en anglais et 70 en français) et 1460 emails légitimes (730 en anglais et 726 en français). Les quelques documents restants sont en espagnol ou dans une langue asiatique. La proportion de documents dans chacune des classes ne semble pas très réaliste. Mais cette valeur est extrêmement variable suivant les utilisateurs et nous n'avons pas cherché à la modifier.

Nous obtenons sur ce corpus seulement 77% de bonne classification avec Rocchio, 93% avec k-PPV et 97% avec SVM. Ces résultats sont inférieurs à ceux rapportés

[3] `http://perso.enst.fr/~buster`

dans les études sur le sujet. C'est sans doute parce que le corpus est constitué à partir des exemples mal classés par SpamBuster. Les exemples présents dans le corpus sont donc a priori des exemples difficiles à classer. Néanmoins, les performances relatives de chaque algorithme restent conformes à la hiérarchie établie. À l'exception de Rocchio, les performances sont toutefois assez élevées. Combinés à d'autres indices non textuels (adresse email émettrice, destinataires, type de documents[4]) les systèmes de filtrage parviennent actuellement à un taux de reconnaissance presque parfait. Il faut néanmoins rester vigilant aux éventuels changements de stratégie des spammeurs (par exemple les messages écrits en modifiant aléatoirement des lettres).

3.3 Détection de points de vue

3.3.1 Présentation du problème

Dans la détection de points de vue, il faut classer les documents en fonction des points de vue exprimés par l'auteur du texte. Un exemple typique est le classement de commentaires de films suivant l'avis positif ou négatif de la critique. Dans cette tâche, les catégories ne sont pas définies par le thème du document. Les termes pertinents pour la classification ne sont plus les mots informatifs (noms communs, noms propres ...) mais tous les autres mots qui pouvent indiquer un avis subjectif : négation, verbes de jugement, adjectifs.

L'idée d'utiliser la CT pour cette tâche est assez récente. Dans [PLV02], les auteurs comparent la catégorisation thématique classique à la catégorisation de sentiments sur le problème des commentaires de films. Turney [Tur02] utilise la classification non supervisée à partir des adjectifs qui portent une orientation sémantique positive ou négative pour séparer les deux types d'avis sur plusieurs tâches : critiques de films, banques, automobiles et destinations de voyage. [BMRR02] évoque l'utilisation de la CT pour détecter l'insatisfaction d'un client dans un courrier électronique.

Pour les trois applications sus-citées, les résultats sont mitigés. Les performances sont toutefois suffisantes pour qu'il soit plus utile d'avoir un système de CT que de faire toute la catégorisation manuellement. Elles restent néanmoins peu élevées si on les compare aux taux habituels en classification thématique. La raison invoquée est que la valeur positive ou négative d'un commentaire n'est pas la somme des parties du texte (un film peut avoir de bons acteurs, de bons décors mais obtenir un avis global défavorable). Les auteurs préconisent donc d'analyser les phrases afin de détecter celles susceptibles d'apporter une information pour la classification.

[4] `http://www.spamassassin.org`

3.3.2 Expérimentations sur le projet *Princip*

Pour traiter ce problème, nous avons étudié la détection de propos racistes sur Internet dans le cadre du projet européen *Princip*[5] (Plateforme pour la Recherche, l'Identification et la Neutralisation des Contenus Illégaux et Préjudiciables sur l'Internet). Ce projet s'inscrit dans l'action *Safer Internet Action Plan* de la Communauté Européenne. Nous avons découpé la tâche de filtrage en deux phases : la détection de la thématique du racisme, qui peut être faite avec des techniques classiques de catégorisation thématique, et la distinction entre avis pro-racistes et avis anti-racistes. Dans cette étude, nous avons considéré que la première phase était réalisée. Il reste alors à différencier les points de vue des auteurs, ce qui est, a priori, une tâche plus difficile.

À l'image du problème des critiques de films, nous supposons que des techniques qui utilisent uniquement le contenu lexical des documents peuvent détecter une différence dans les deux types de textes. Les mots discriminants ne sont plus les noms communs qui décrivent plutôt le thème du document mais les termes subjectifs (adjectifs, verbes). Les algorithmes de CT vont faire eux-mêmes la sélection des termes utiles.

Notre problème semble plus facile que celui des critiques de film car une personne ne peut pas ici être auteur de textes dans les deux classes (un auteur ne peut pas à la fois être pro et anti-raciste). Le classifieur peut donc chercher à reconnaître le style des différents auteurs. De plus, bien que la tâche ne soit pas thématique, les documents pro-racistes et anti-racistes ne parlent souvent pas des mêmes sujets. Les exemples des figures 3.2 et 3.3 sont symptomatiques : beaucoup de documents pro-racistes sont en fait la description d'un évènement, alors que les sujets des documents anti-racistes sont souvent plus généraux. Il doit donc être possible d'isoler l'ensemble des thématiques pro-racistes et des thématiques anti-racistes comme nous l'avons fait par exemple pour le Spam. Le problème devient à nouveau celui des catégories multi-thématiques.

Pour pouvoir faire des expériences, nous avons construit un corpus de pages web pro et anti-racistes. Le processus de construction commence par un sélection de quelques mots-clés qui nous permettent de récupérer des pages web potentiellement intéressantes. La lecture de ces pages et surtout l'analyse des liens (grâce à un effet communautaire inter-sites) permet de trouver de nouveaux mots-clés pour lancer de nouvelles requêtes et ainsi de suite. La constitution de ce corpus a été effectuée au cours de la première année de travail du projet Princip. Le corpus sur lequel nous avons travaillé (qui n'est pas le corpus final) contient 742 pages dont 287 pro-racistes et 455 anti-racistes tirées de 121 sites différents (dont quatre majoritaires avec 67% des pages). Pour les expérimentations, nous avons utilisé le principe de validation croisée avec 10 sous-ensembles (voir la section 2.6.1). Les résultats que nous obtenons sur ce corpus sont présentés dans le tableau 3.1.

[5] `http://www.princip.net`

Numéro 46 - Mai 1997 ——————–
Pour chaque numéro, nous plaçons "en ligne" l'éditorial et le dossier.
Edito par Ras l'Front
Votez, éliminez ! Pas de FN à l'Assemblée
Le combat contre le Front national connaît à n'en pas douter une vigueur nouvelle dont nous ne pouvons que nous féliciter. Par dizaines de milliers, nous avons manifesté de Grenoble à Strasbourg, de Toulouse à Vitrolles. Et nous comptons bien continuer à accueillir comme il se doit les Le Pen, Mégret, Gollnisch et autres, tout au long de la campagne éclair de ces législatives. Car, si d'aucuns débattaient d'une éventuelle dissolution du Front national ou de son bras armé la DPS, c'est l'Assemblée nationale que le Président a décidé de dissoudre. Soit, ce sera donc là l'occasion de sanctionner ceux qui, croyant renforcer leur base électorale en empruntant concepts et idées au FN, ont mené la politique du pire, confère les lois Debré. Ce sera aussi l'occasion de mesurer l'engagement de ceux qui, multipliant les déclarations anti-Le Pen traînent la patte quand il s'agit d'énoncer haut et clair les valeurs sur lesquelles nous ne saurions reculer.
Et dans moins de trois semaines, nous saurons (en partie tout du moins) ce qu'il en est de l'audience du Front national, audience que nous redoutons de voir se confirmer. Et pourquoi pas voir un député (ou plusieurs) fasciste faire ses premiers pas à l'Assemblée. En 1986, s'ils étaient alors plusieurs dizaines, rappelons-nous qu'ils bénéficiaient du scrutin proportionnel. Onze ans plus tard, l'entrée d'un député FN se fera au scrutin majoritaire, tout comme le gain de Vitrolles par Mégret. Et comme le "hasard" fait bien les choses, ce sera probablement le même qui inscrira son nom au sein de l'hémicycle (monsieur à la place de madame, mais quelle différence ?).
Cette montée en puissance du Front national continuera donc de poser problème. Mais il y aussi la possibilité de voir le Front national ne pas envoyer de député à la Chambre, progresser sans élu ou même se "contenter" d'un score identique aux dernières législatives. Nombreux seront alors ceux qui parleront du début du déclin, qui penseront que ce scrutin de mai-juin invalide les municipales partielles de Vitrolles. Dans cette dernière hypothèse, le combat contre le Front - un certain combat - pourrait connaître alors une baisse d'intensité.
Le travail de terrain, véritable travail de sape, opéré par le Front national, reprendra avec plus de vigueur, n'en doutons pas. Dopé par un porte-voix parlementaire ou par la conviction que décidément, le jeu démocratique ne leur convient pas. Dans un cas comme dans l'autre, il faudra que tous les antifascistes restent mobilisés, vigilants, actifs.
Le réseau Ras l'front qui n'a jamais autant progressé que ces derniers mois, voyant le nombre de ses collectifs croître, le journal multiplier ses ventes, restera plus que jamais mobilisé. Disponible aussi pour tous ceux qui ne se satisferont pas d'un résultat électoral qui risque, de toutes les façons, de ne pas correspondre à l'état réel de la société et à l'influence dramatique du Front national sur celle-ci.
Alors, à la fin du mois, votez, éliminez et ensemble, tous ensemble continuons de construire la contre-offensive au fascisme et au racisme.
[...]

FIG. 3.2 – Exemple d'une page anti-raciste

Lieu : Rue des Rigoles
Type d'agression : Vol à l'arraché
J'allais m'endormir lorsqu'un hurlement de femme est monté de la rue jusque chez moi. Je me suis levé, je me suis précipité à la fenêtre. Une femme gisait sur le trottoir. On venait de lui voler son sac et comme elle avait résisté un peu, l'agresseur l'avait frappée et poussée à terre.
J'ai enfilé quelques vêtements, passé mes chaussures sans chaussettes. Je suis sorti précipitamment de mon appartement et j'ai dévalé les escaliers plutôt que d'attendre l'ascenseur. Lorsque je suis arrivé dans le hall de mon immeuble, le voleur était déjà loin et plusieurs personnes étaient venues entourer la victime, une française, qui, complètement sonnée, restait assise au bord du caniveau. Je n'aurais rien pu faire pour cette femme et je suis remonté chez moi .
En toute sincérité, j'ignore l'origine ethnique des agresseurs.

FIG. 3.3 – Exemple d'une page pro-raciste

	Rocchio	**10-PPV**	**SVM**
Uniquement le texte	0.89	0.94	0.95
Avec le code HTML	0.94	0.95	0.96

TAB. 3.1 – Résultats sur le corpus Princip

Les taux de classification sont particulièrement élevés. Il est même possible d'obtenir plus de 99% de bonne reconnaissance si l'on accepte de ne pas classer 20% de documents. Ceci montre que la détection d'opinion dans ce cas est envisageable avec des techniques purement lexicales. L'analyse des erreurs met en évidence les mêmes phénomènes que ceux de [PLV02] : la catégorie d'un document n'est pas égale à la catégorie moyenne de chaque phrase, il suffit qu'une phrase soit considérée comme pro-raciste pour que le document dans son ensemble doive être considéré comme pro-raciste. Or les algorithmes actuels, qui utilisent comme représentation du texte la somme de tous les mots présents dans le texte, ne sont pas capables de se focaliser sur les phrases les plus discriminantes. La deuxième raison qui explique les erreurs obtenues provient de l'aspect parfois ironique ou euphémistique des propos que les algorithmes ont du mal à appréhender.

Afin de répondre au premier problème, nous avons découpé tous les documents en plusieurs segments de même taille (100 mots pour chacun d'entre eux) et essayé de catégoriser chacun des segments indépendamment. Chaque document est classé comme pro-raciste si au moins un des segments est classé comme pro-raciste. Pour que la base d'apprentissage soit similaire au corpus de test, nous avons également découpé les documents de l'apprentissage en segments. Nous connaissons la catégorie de chaque document mais pas celle de chaque segment. Pour apprendre cette information, nous avons utilisé un algorithme EM [DLR77]. Nous initialisons la catégorie des segments suivant la catégorie du document. Un classifieur est appris et testé sur ces segments. La catégorie des

segments est alors attribuée en fonction du résultat du classifieur et ainsi de suite jusqu'à stabilité des étiquettes. Malheureusement, aucune de ces techniques ne s'est avérée intéressante : les performances sont dans tous les cas inférieures au système simple. Les segments que nous avons utilisés sont découpés de façon rigide tous les 100 mots, ce qui ne correspond pas forcément à un découpage cohérent du document. Les segments que l'on trouve sont donc très bruités (d'autant plus qu'ils contiennent peu de mots) et la classification n'est pas performante.

Nous avons réalisé un test supplémentaire avec prise en compte non seulement du texte des documents mais également de tout le code source HTML de chaque page. Les performances sont alors légèrement améliorées. L'analyse des mots discriminants a permis de mettre en évidence dans le corpus de sites pro-racistes l'importance des sites qui couvrent les faits divers (ils comprennent beaucoup de dates récentes), l'utilisation de polices de caractères particulières et l'abondance d'images et de logos. Ceci montre qu'il est utile de fournir aux algorithmes le plus d'informations péritextuelles possibles. Ces informations peuvent être discriminantes et les intégrer dans la représentation du document permet aux classifieurs de les utiliser judicieusement. C'est également le cas avec le Spam où l'utilisation des en-têtes permet en général d'améliorer les taux de reconnaissance. Une analyse plus approfondie des résultats a été présentée dans un article que nous avons publié à la conférence TALN 2003 : "Application d'algorithmes de classification automatique pour la détection de contenus racistes sur l'Internet" [VGV03].

3.4 le Mail Center

3.4.1 Présentation du problème

Le principe du Mail Center, que l'on peut comparer à celui du Call Center pour le téléphone, est de centraliser les correspondances électroniques des clients dans un lieu spécialisé. Cette centralisation permet une gestion optimisée du flux des correspondances par une meilleure répartition des tâches, une administration simplifiée, ainsi que la possibilité d'utiliser des outils supplémentaires pour accélérer les traitements et mieux satisfaire les clients.

Parmi ces outils, on trouve les modules de réponse automatique ou semi-automatique. Leila Kosseim [Kos00] a réalisé une synthèse de tous les produits commerciaux disponibles pour effectuer cette tâche. Il s'agit généralement de restreindre l'utilisateur dans la formulation de sa question (à l'aide d'un formulaire web par exemple) afin que l'on puisse facilement détecter son contenu. Le système peut éventuellement proposer des raffinements pour personnaliser les réponses (avec le nom du client ou le numéro de commande par exemple). Nous nous sommes interessés au cas, plus ardu, où les questions ne sont pas restreintes. L'utilisateur a la possibilité d'écrire n'importe quel texte contenant

une question ou une demande. Le système doit sélectionner une réponse parmi un ensemble de réponses-types. Une fois que la réponse est choisie, elle peut être validée par un opérateur avant l'envoi définitif. Si cette validation est obligatoire, le système est dit semi-automatique (comme la correspondance concerne généralement une entreprise et ses clients, il est délicat de laisser une machine répondre automatiquement car la réponse choisie n'est pas forcément correcte et on préfère souvent une validation humaine). Les courriers auxquels on peut répondre par une réponse-type peuvent représenter jusqu'à 50 % des courriers entrants [BSA00, VY01]. Ces systèmes semi-automatiques permettent donc un gain de temps important.

La sélection d'une réponse-type est un problème de CT : les catégories correspondent aux différentes réponses-types. Le corpus d'apprentissage est constitué de l'ensemble des emails pour lesquels un opérateur a déjà validé la réponse. Dans cette application, le nombre de classes est souvent important (plus d'une centaine en général), ce qui rend la tâche d'autant plus difficile. Le corpus peut, comme dans le cas du Spam, mêler plusieurs langues (mais cette fois, les catégories sont monolingues : un email écrit dans une langue doit trouver sa réponse dans la même langue). La figure 3.4 présente quelques exemples de courriers reçus dans un Mail Center.

Quelques réponses-types peuvent se différencier par leur thème, mais ce n'est pas le cas pour toutes. Plusieurs réponses peuvent porter sur le même thème, voire dépendre d'informations extérieures à la question (une demande concernant l'état d'une commande par exemple). Des mots généralement supprimés comme les indices de négation, peuvent être ici très importants. Comme pour la détection des opinions, la question à laquelle il faut répondre est souvent concentrée autour d'une phrase et non pas dans le document entier. Malgré ces difficultés, nous faisons l'hypothèse que les algorithmes de CT avec une représentation uniquement lexicale des textes sont à même d'effectuer cette tâche.

3.4.2 Expérimentations

Nous avons développé un module de ce type dans le logiciel Akio Mail Center de la société Akio Software[6]. Nous avons pu obtenir des données réelles de la part d'entreprises utilisatrices de ce logiciel, ce qui nous a permis d'effectuer des études plus approfondies. Nous avons travaillé essentiellement avec un corpus de 2393 documents, inégalement répartis en 40 classes (les 10 classes les plus fréquentes réunissent 70% des documents). Les résultats ont été présentés dans un article publié à la conférence ASMDA 2001 : "Semi-automatic response in a Mail Center" [VY01]. Ils sont conformes à ceux obtenus dans [BSA00]. Les algorithmes trouvent la bonne réponse comme premier choix pour un peu plus de 50% des emails et SVM est l'algorithme le plus performant. Cela n'est

[6] `http://www.akio-software.com`

je n'ai pas pu me contacter depuis mon inscription et on me dit a chaque fois que je veux me reinscrire qu'il ai deja utilise alors que je suis sur de m'avoir inscrit.
Je n'ai pas reçus votre e-mail pour l'ouverture de conte
Quand je renvoi le message de confirmation j'ai comme erreur : Erreur interne. Que faire. Merci
Je viens de changer d'adresse e-mail, voici la nouvelle : ...@....fr
votre site nécessite les cookies, mais je n'arrive pas a l'utiliser avec un système ... où j'ai complètement supprimé Internet Explorer pour mettre Netscape à la place, alors que plein d'autres sites n'ont aucun problème pour mettre un cookie à mon Netscape (j'ai essayé les versions 4.5, 4.7 et 6 qui ont toutes le même problème avec ...) Merci de vérifier comment vos cookies sont envoyés... Sinon je vais devoir tout réinstaller ou bien me contenter d'utiliser Netscape sur une machine Unix.

FIG. 3.4 – Exemples de courriers reçus par un Mail Center (avec les fautes d'orthographe...)

plus vrai dès que l'on prend en compte le fait que le système fonctionne en mode semi-automatique. Il peut être utile de proposer plusieurs classes afin que l'opérateur choisisse la plus adaptée. Les mesures de performance doivent alors être modifiées pour refléter ce type d'interaction. Nous en étudierons les conséquences dans la section 4.4.

3.5 Classement de dépêches d'informations suivant l'évènement générateur

3.5.1 Description du projet TDT

TDT (Topic Detection and Tracking) [7] est un projet de recherche qui fait partie du programme *Translingual Information Detection, Extraction, and Summarization* (TIDES) sponsorisé par l'agence DARPA (Defense Advanced Research Projects Agency). Il s'agit dans ce projet de développer des techniques automatiques pour localiser des informations portant sur un même évènement à partir de flux de données hétérogènes provenant de journaux (papier ou en ligne) et d'émissions audiovisuelles dans plusieurs langues.

Ce projet se différencie surtout par le type des catégories recherchées. Elles ne sont pas définies par des thèmes génériques mais par l'évènement qui a généré la nouvelle (dans la terminologie de TDT, ces catégories sont appelées *topic*). Par exemple, pour un texte qui parle du tremblement de terre survenu à Kobe en 1995, les catégories thé-

[7] `http://www.nist.gov/speech/tests/tdt/`

matiques classiques seraient, suivant le degré de classification souhaité : environnement, catastrophes naturelles ou tremblement de terre. Dans le cadre de TDT, la catégorie correspondante est l'ensemble des nouvelles qui ont un rapport avec le tremblement de terre précis de Kobe : donc la description de l'évènement lui-même mais également tout ce qui concerne ses conséquences comme les pannes d'infrastructure ou le coût des réparations. Ces dernières nouvelles peuvent ne contenir que peu de mots lexicalement proches du thème des tremblements de terre. Comme pour les applications présentées précédemment, les catégories ne sont donc pas thématiques. La figure 3.5 présente deux articles extraits du corpus TDT.

Le problème que l'on cherche à résoudre dans le projet TDT a été formalisé en cinq tâches distinctes sur lesquelles les systèmes automatiques peuvent être évalués quantitativement et objectivement. Ces cinq tâches sont :

- Segmentation des nouvelles : détecter les changements de sujets pour segmenter le flux d'information en nouvelles cohérentes.
- Classification supervisée (Topic Tracking) : Détecter les nouvelles similaires à un ensemble d'articles présélectionnés qui se rapportent à un même évènement.
- Classification non supervisée (Topic Detection) : Grouper les nouvelles suivant des ensembles qui concernent un même évènement.
- Détection de nouveauté (First Story Detection) : Détecter si une nouvelle concerne un évènement déjà rencontré ou non.
- Détection de liens (Link Detection) : Décider si deux nouvelles parlent du même évènement ou non.

Pour réaliser des évaluations importantes, plusieurs corpus ont été réalisés (chacun augmentant le précédent de façon à accroître la quantité de données). Nous avons utilisé le corpus TDT3[8], qui contient des données collectées sur une période de 3 mois (de octobre à décembre 1998) de 11 sources de nouvelles (8 sources en anglais et 3 sources en mandarin) qui proviennent d'émissions de télévision (4 sources), de radio (3 sources) ou textuelles (4 sources) pour un total de 43 600 documents. 60 catégories correspondant à des évènements ayant eu lieu au cours de ces trois mois ont été prédéfinies. Tous les articles ont été étiquetés suivant leur pertinence pour les 60 sujets. 3600 articles ont été jugés pertinents pour au moins une des catégories.

3.5.2 Expérimentations sur la tâche de Topic Tracking

Nous nous sommes interessés au cours de cette thèse uniquement à la deuxième tâche, c'est-à-dire la tâche de catégorisation. La tâche Topic Tracking considère un contexte de filtrage dans lequel il faut décider pour chaque document s'il fait partie ou non de chaque catégorie. Ce principe correspond à une application où les utilisateurs peuvent

[8] `http://www.ldc.upenn.edu/Projects/TDT3/`

Traditionally on Christmas Day, there are some professional basketball games here in the United States, however, breaking with tradition this year, there will be no hardwood play. At least in the pro ranks. National Basketball Association commissioner, David Stern, has said there will be no season if a labor agreement between team owners and players is not reached by January 7. Now Stern has scheduled a board of governor's meeting for that day when he will ask 29 owners for permission to cancel the season. All he needs is a simple majority. Talks on a new deal between the owners and players' union broke off two weeks ago. Stern says there are still serious disagreements that led to the player lockout July 1. This is not about the owners expecting the players to cave and we wanted a strong union. We wanted the players to understand that the system under which the owners have operated is just not one that makes economic sense. The dispute centers on how to split what would have been $2 billion in revenue this season. As well as salary caps for the highest paid players. Two months of the season have already been lost as well as the all-star game, players have lost millions of dollars in salaries.

Dr. Jack Kevorkian is scheduled to appear before a preliminary hearing december 9th for his upcoming trial in Michigan. Today, he was charged with first-degree murder for his role in the videotaped death of a terminally ill man. "60 minutes" broadcast parts of the tape Sunday. A magistrate released Kevorkian after he was arraigned when he agreed not to take part in any suicide or euthanasia cases.

FIG. 3.5 – Exemples d'articles du corpus TDT

sélectionner quelques nouvelles et indiquer au système : "Je souhaite recevoir toutes les informations qui traitent de cet évènement". Dans le cadre de cette thèse, nous nous sommes plus particulièrement intéressés par les problèmes de routage. Ce contexte peut intervenir dans le cadre d'une application qui contrôlerait l'ensemble des nouvelles reçues et qui doit systématiquement sélectionner pour chaque nouvelle l'évènement correspondant parmi l'ensemble des évènements potentiels. Pour nous mettre dans ce cadre, nous avons donc réalisé des expériences en considérant que les seuls évènements possibles étaient les 60 catégories prédéfinies et nous avons supprimé tous les documents qui ne traitaient pas de ces sujets.

Nous avons réalisés des tests avec les algorithmes Rocchio, k-PPV et SVM. Comme avec le corpus Princip (section 3.3.2), les classifieurs sont particulièrement performants et font moins de 5% d'erreur (95.6% de taux de performance). Il n'y a pas de différence significative entre les différents algorithmes. Il existe essentiellement deux raisons pour expliquer les bonnes performances des classifieurs standards sur ce corpus.

Premièrement, nous avons testé le corpus dans un contexte de routage. Pourtant, le corpus a été réalisé en prenant en compte un contexte de filtrage. Toutes les catégories possibles dans un tel corpus n'ont donc pas été déterminées mais seul un échantillon de 60 d'entre elles a été utilisé. Les catégories qui ont été choisies sont souvent thématiquement

éloignées les unes des autres. Ainsi, l'une des catégories concerne le passage de l'ouragan Mitch dans les Caraïbes. Mais il n'y a pas d'autre classe parmi les 60 qui parle d'ouragan. Les classifieurs peuvent donc assigner la catégorie qui correspond à l'évènement de l'ouragan Mitch (qui est non thématique par définition) à tous les documents qui traitent du thème des tremblements de terre (qui lui est thématique). On peut donc supposer que les performances des différents algorithmes baisseront lorsqu'ils seront utilisés sur des corpus avec beaucoup plus de classes qui sont plus proches les unes des autres. Pour pallier cette augmentation de la difficulté, il est possible d'ajouter un module de gestion temporelle des catégories. En effet, il est peu probable que deux évènements du même type apparaissent au même moment. L'année où a sévi l'ouragan Mitch, tous les documents sur le thème des ouragans portent sur cet ouragan. L'année suivante, les documents de ce thème concerneront un autre évènement (un autre ouragan par exemple). À un instant donné, il existe en général peu d'évènements correspondant à chaque thème.

Deuxièmement, les catégories ne sont effectivement pas thématiques. Les termes discriminants ne sont pas les noms communs porteurs de sens (qui déterminent souvent une thématique), mais plutôt les entités nommées qui décrivent très rapidement les évènements dont l'on parle. Bien que non thématiques, les catégories sont très facilement reconnaissables avec les informations lexicales. Il suffit de trouver le terme "Mitch" pour connaître automatiquement la catégorie du document. De plus, dans ce type de texte, les lecteurs (ou auditeurs) doivent pouvoir appréhender le plus rapidement possible de quel évènement est en train de parler une nouvelle. Les mots-clés qui permettent de faire cette reconnaissance sont donc souvent mis en évidence au début du texte de la nouvelle. Le terme "Mitch" suffit à classer un document dans la catégorie et ce terme sera systématiquement prononcé parce qu'il permet aux lecteurs de reconnaître très rapidement de quoi parle le document.

Le problème du classement des dépêches suivant l'évènement générateur est donc un problème typiquement non thématique. Pourtant, les techniques classiques de CT à partir d'une représentation lexicale du texte suffisent largement pour répondre à ce problème.

3.6 Conclusion

Nous avons présenté dans ce chapitre quatre nouvelles applications pour la CT qui ont toutes la particularité de chercher à classer les documents dans des catégories qui ne sont pas définies par une thématique précise. Malgré cette caractéristique, les algorithmes de CT peuvent être parfaitement adaptés à ces problèmes. C'est le cas pour la détection de propos raciste comme nous l'avons étudié au cours du projet Princip ou le classement de dépêches suivant l'évènement générateur comme proposé par le projet TDT.

Ces nouvelles applications impliquent souvent des caractéristiques particulières auxquelles les techniques de CT doivent s'adapter. Les classes sont très bruitées (le corpus d'apprentissage contient beaucoup de documents mal étiquetés) et les classifieurs peuvent être utilisés dans un cadre semi-automatique avec une supervision humaine.

Nous verrons dans le chapitre suivant, à travers quatre exemples, que les caractéristiques des corpus présentées au cours de ce chapitre modifient de façon importante les performances des algorithmes classiques et qu'il est important de prendre en compte le contexte applicatif dans lequel le système de CT est mis en place.

CHAPITRE

4

Des algorithmes spécifiques pour les nouvelles applications

Sommaire du chapitre

4.1 Introduction

NOUS AVONS vu dans le troisième chapitre plusieurs exemples de nouvelles applications qui peuvent bénéficier des techniques de catégorisation de textes (CT). Dans ces applications, les systèmes de CT sont utilisés dans un contexte légèrement différent de celui pour lequel ils étaient prévus à l'origine. La plupart des techniques sont néanmoins suffisament générales pour pouvoir être appliquées à de nouveaux contextes

sans modification. C'est le cas, par exemple, du modèle vectoriel, utilisé dans beaucoup d'applications qui manipulent des documents textuels comme l'étude de textes anciens, l'analyse de questions ouvertes... C'est aussi le cas des algorithmes de classification qui sont employés dans des domaines comme la biologie ou la reconnaissance d'images.

Néanmoins, certaines techniques peuvent voir leur comportement changer suivant l'application et s'avérer peu performantes, voire inutiles lorsqu'on les utilise dans un cadre différent de celui pour lequel elles ont été conçues. Il est donc important, lorsque l'on souhaite mettre en place un système de CT, de s'assurer que tous les modules se comporteront correctement par rapport à la tâche fixée par l'application. Celles que nous avons décrites dans le chapitre précédent présentent plusieurs particularités dont nous allons chercher à montrer l'importance :

- À l'exception de la tâche de détection du Spam, qui est typiquement un problème de filtrage, les autres applications se modélisent comme du routage,
- Ne pouvant fournir de garantie sur leur catégorisation, les classifieurs sont parfois utilisés en mode semi-automatique,
- Les corpus d'apprentissage étant constitués avec moins de soin que les corpus de référence sur lesquels sont généralement testés les nouveaux algorithmes, ils contiennent souvent plus de documents mal étiquetés, donc plus de bruit.

Nous présentons dans ce chapitre une analyse de plusieurs algorithmes ainsi que des expérimentations sur divers corpus pour mettre en évidence l'importance de ces trois facteurs. Nous commençons par étudier l'intérêt de la racinisation des mots (stemming) suivant que la tâche soit de la Recherche d'Informations (RI) ou de la Catégorisation de Textes (CT) (section 4.2). Nous analysons ensuite l'utilité du Query Zoning pour améliorer Rocchio dans un cadre de filtrage ou de routage (section 4.3). Nous présentons dans un troisième temps une nouvelle mesure de performance nommée perf-x qui prend en compte une utilisation semi-automatique des classifieurs. Les algorithmes Rocchio, k-PPV et SVM sont testés sur cette nouvelle mesure et nous proposons une explication pour les résultats obtenus (section 4.4). Enfin, nous avons réalisé une expérience à partir d'un corpus réel que nous bruitons artificiellement pour visualiser les différences de comportement entre Rocchio, k-PPV et SVM face au bruit (section 4.5).

4.2 Recherche d'Informations et Catégorisation

4.2.1 Comparaison des deux types de tâches

La RI et la CT partagent un grand nombre de points de vue et de méthodes. Dans les deux cas, il s'agit de sélectionner les documents qui comportent une information pertinente pour l'utilisateur. Il est même possible de représenter le problème de RI sous la forme d'une tâche de CT. À chaque requête correspond un problème de CT constitué

de deux classes : la catégorie des « documents pertinents » et celle des « documents non pertinents ». Il s'agit alors de classer l'ensemble des documents dans une de ces deux classes.

La différence principale entre les deux approches réside dans le type d'informations recherché par l'utilisateur : la RI est utilisée pour des demandes *ponctuelles* alors que la CT modélise un intérêt *à long terme*. Cette différence implique des variations sur le type et la quantité d'information que l'utilisateur peut fournir au système dans les deux modes de recherche. En RI, l'utilisateur n'a pas le temps de décrire très précisément sa requête, il se contente de fournir quelques mots-clés (typiquement de un à cinq). En CT, le besoin de catégoriser peut perdurer plusieurs mois ou même sans horizon précis, le système a donc la possibilité d'utiliser toute l'information contenue dans l'ensemble des documents déjà catégorisés. La quantité d'information disponible est bien plus importante. Cette différence s'atténue lorsque le système de RI interagit avec l'utilisateur pour préciser les requêtes (feedback : voir section 2.3.1). Dans ce cas, l'information utilisée comprend la requête initiale, ainsi que les documents pour lesquels l'utilisateur a fourni un jugement. Le feedback utilise donc également la notion de documents catégorisés, ce qui rend son fonctionnement très proche de celui de la CT. Les algorithmes développés pour l'exploiter (Rocchio et Naïve Bayes) ont d'ailleurs été appliqués avec succès en CT. Néanmoins, les deux approches ne sont pas toujours comparables en raison de la différence du nombre de documents dans le corpus d'apprentissage : lors d'une recherche avec feedback, l'utilisateur fournit un jugement pour quelques documents (de un à quelques dizaines), alors qu'en CT le corpus peut en contenir plusieurs milliers. C'est pourquoi les algorithmes de CT peuvent utiliser une représentation plus complexe sans risquer le sur-apprentissage des paramètres, ce que ne peuvent pas faire les algorithmes de feedback.

4.2.2 Utilité de la racinisation

Pour illustrer cette différence, nous avons comparé l'utilisation d'un module de racinisation dans un cadre de RI et de CT. La racinisation présentée dans la section 2.2.1 a deux fonctions : (i) diminuer le nombre de termes à indexer et donc les temps de calcul, et (ii) améliorer les performances de rappel en permettant une correspondance entre des mots différents (mais de même famille). Par exemple, lorsque la requête contient le mot « catégorisation », celui-ci est simplifié en « catégoris », ce qui permet une correspondance avec les mots « catégoriseur » et « catégoriser » qui pourraient se trouver dans un document. Il permet malheureusement aussi de mettre en relation des mots qui ne devraient pas l'être comme « racine » et « raciner » qui sont deux termes de la même famille lexicale mais de significations complètement différentes. Si l'utilisation de la racinisation est assez controversée, elle semble globalement permettre une très légère

amélioration des performances en RI (entre 0 et 2%) [Por80].

En CT, nous pensons que la deuxième fonction de la racinisation est superflue en raison du grand nombre de documents dans le corpus d'apprentissage. Si le texte à classer contient le mot «catégorisation », il n'est pas nécessaire de faire une correspondance avec des documents contenant les mots « catégoriseur » ou « catégoriser ». Il existera certainement (si le corpus est suffisamment grand) d'autres documents contenant exactement le mot « catégorisation ». La correspondance se fera avec ces documents et la classification sera correcte même si les documents contenant les autres mots de la même famille n'ont pas été sélectionnés. Comme la racinisation permet aussi des correspondances inexactes, nous pensons que son emploi peut être plus néfaste qu'avantageux en CT.

Pour valider cette analyse, nous avons mesuré expérimentalement l'apport de la racinisation dans une tâche de CT. Nous avons utilisé le corpus 20 Newsgroups (voir section 2.6.2) avec l'algorithme 1-PPV : pour chaque document de test, on sélectionne l'exemple le plus proche du corpus d'apprentissage et on attribue la classe correspondante. L'algorithme de racinisation utilisé est celui de Porter [Por80] pour l'anglais. La performance est mesurée par le taux de classification correcte. Les résultats obtenus sont de 81% sans racinisation et 75% avec racinisation. Alors que en RI, les différentes expérimentations réalisées au cours des conférences TREC [NDa] montrent une amélioration faible de l'ordre de 1%, nous obtenons en CT une baisse de 6% ! Cette contre-performance a également été mise en évidence dans les travaux de Leopold avec l'algorithme SVM [LK02]. La racinisation est donc plutôt utile en RI mais inutile en CT, ce qui confirme l'analyse que nous venons d'effectuer.

4.3 Filtrage et routage

4.3.1 Description

Il existe au sein de la CT différents types de tâches. Les deux principales sont le filtrage et le routage présentés à la section 2.1. Dans le filtrage, le système doit déterminer si le document est pertinent ou non pour chaque catégorie. L'affectation dans une classe est indépendante des décisions sur les autres classes. Le problème peut être transformé en plusieurs sous-problèmes biclasses indépendants. C'est le type de tâche le plus courant (l'application de référence de classement de dépêches de presse est de ce type). Dans le routage au contraire, le système doit choisir une seule classe. Il ne s'agit pas de trouver toutes les classes pertinentes mais seulement la classe la plus pertinente. Les classes sont donc en opposition directe. Cette fois, le problème est intrinsèquement multiclasse. Les nouvelles applications présentées au chapitre précédent sont plutôt de ce type.

Lorsque les algorithmes de classification fournissent une valeur de confiance pour la

prédiction d'appartenance d'un document à une classe (et c'est le cas de la majorité des algorithmes : la mesure de similarité pour Rocchio, le pourcentage de votes pour k-PPV, la distance à la surface frontière pour SVM), il est possible d'utiliser ces valeurs pour du filtrage ou du routage. En filtrage, la valeur de confiance est comparée à un seuil fixé préalablement et le document est affecté à toutes les classes dont la valeur dépasse ce seuil. En routage, le document est assigné à la classe ayant la plus grande valeur de confiance.

En filtrage, chaque décision d'affectation à une classe met en concurrence une catégorie face au reste du corpus. Comme il est difficile de créer un profil correspondant à l'ensemble du corpus (qui doit donc comprendre l'ensemble des thèmes évoqués par les documents), les méthodes de filtrage consistent en général à créer un profil pour la catégorie recherchée (il n'y a pas de profil pour la classe générique) et faire en sorte que ce profil corresponde le mieux possible à la thématique. C'est un cadre qui "convient" bien aux modèles génératifs. Dans ces algorithmes, un modèle (souvent probabiliste) est proposé a priori. Il permet d'expliquer comment les documents sont *générés*. Tous les paramètres de ce modèle sont estimés à partir des documents du corpus. Par exemple, avec l'estimateur du maximum de vraisemblance, les valeurs des paramètres sont calculées de façon à ce que la probabilité du corpus (c'est-à-dire la probabilité que le processus génère exactement l'ensemble des documents du corpus) soit la plus élévée possible. En routage, la description des classes n'est pas nécessaire, il suffit que la séparation des classes soit correcte. On peut s'attendre à ce que les modèles discriminants (où seules les frontières entre classes sont importantes) soient plus performants dans ce cas. Le principe sous-jacent à ces deux types de classifieurs (génératifs et discriminants) est illustré dans la figure 4.1. Les deux modèles ont été comparés dans un article de Jordan et Ng [NJ01] : les modèles discriminants ont souvent de meilleurs taux de classification asymptotique (i.e. avec un nombre infini d'exemples d'apprentissage) mais mettent plus de temps à l'atteindre que les modèles génératifs et ils sont souvent computationnellement plus lourds. Par ailleurs, les algorithmes génératifs peuvent généralement expliquer les modèles de classe qui ont été calculés à des experts humains (par exemple en fournissant les mots les plus pertinents de chaque classe) alors que les algorithmes discriminants sont souvent incapables de fournir des explications intelligibles aux experts humains pour expliquer les raisons de chaque décision de classification (le classifieur fonctionne comme une boîte noire).

Il est possible d'utiliser un algorithme génératif pour faire du routage (par exemple Rocchio ou Naïve Bayes). Dans ce cas, les algorithmes choisissent la catégorie qui correspond le plus au document. En revanche, il est plus difficile d'utiliser un algorithme discriminant (par exemple k-PPV ou SVM) pour faire du filtrage. Dans ce dernier cas, les deux classes sont complètement assymétriques (une catégorie correspond à un thème

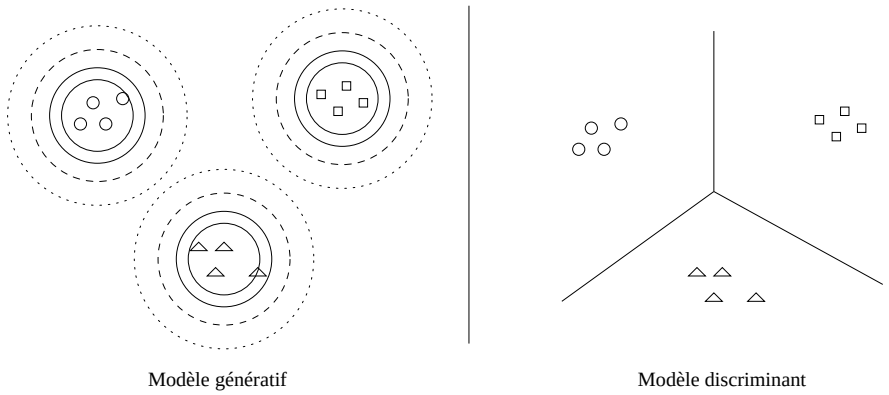


FIG. 4.1 – Illustration des modèles génératifs et discriminants : les algorithmes génératifs tentent de décrire les classes (par exemple ici sous la forme de gaussienne) alors que les algorithmes discriminants ne s'intéressent qu'à la modélisation des frontières (par exemple ici des surfaces séparatrices sous la forme d'hyperplans).

alors que l'autre correspond à l'ensemble des autres thèmes). Un document qui parle d'un thème inconnu doit donc être mis dans la deuxième classe bien qu'il ne corresponde à aucune des deux. Les algorithmes discriminants sont souvent gênés par cette dissymétrie. Par ailleurs, chaque algorithme optimise une mesure particulière qui est plus ou moins adaptée aux différents cas de figure. Un algorithme performant en filtrage ne le sera pas forcément en routage et inversement.

4.3.2 Query Zoning

Nous avons réalisé une expérience pour comparer l'utilité du Query Zoning [SMB97] pour l'algorithme Rocchio [Roc71] d'une part en routage et d'autre part en filtrage. Rappelons que la formule de Rocchio pour calculer le profil d'une classe est :

$$R_{jk} = \max \left\{ \begin{array}{l} \frac{t}{|c_j|} \sum_{d_i \in c_j} d_{ik} - \frac{1-t}{|\overline{c_j}|} \sum_{d_i \notin c_j} d_{ik} \\ 0 \end{array} \right. \tag{4.1}$$

Dans cette formule, le prototype est calculé à partir des exemples positifs et des exemples négatifs. Ceci permet de ne faire entrer dans le profil que les termes les plus significatifs. Un terme qui n'est pas plus utilisé dans une classe que dans le reste du corpus n'est pas spécifique à cette classe et son poids doit donc être proche de 0. Le paramètre t permet de donner plus ou moins de poids aux éléments positifs et négatifs. Les valeurs habituelles sont 0.5, 0.8 ou 1. Dans ce dernier cas, les documents négatifs ont un poids nul et le profil est exactement le barycentre des exemples de la classe.

Dans cette formule tous les exemples négatifs ont le même poids, qu'ils soient d'un thème proche de celui de la classe ou d'un thème très éloigné. Par exemple, si la ca-

tégorie recherchée concerne les logiciels de CT, les documents négatifs, qu'ils parlent d'économie ou d'informatique (mais pas de logiciels de CT) ont le même poids. Affecter une valeur négative aux termes qui concernent l'économie est pourtant inutile puiqu'ils n'ont de toute façon aucune chance d'apparaître dans un document qui parle de catégorisation. En revanche, les termes comme *ordinateur*, *interface* sont du domaine de l'informatique. Ils sont susceptibles d'apparaître dans un document sur la CT mais ne sont pas spécifiques de ce domaine. Il est donc utile de faire baisser leur contribution dans le profil. Les éléments négatifs d'un thème éloigné sont donc moins utiles pour la description du profil que les éléments proches.

Cette idée a été mise en application dans le Query Zoning [SMB97]. Au lieu de prendre l'ensemble des documents du corpus comme éléments négatifs, une zone autour du profil de la classe est calculée et seuls les documents de cette zone sont pris en compte comme exemples négatifs. Il y a plusieurs façons de définir cette zone mais les comportements sont sensiblement identiques dans tous les cas. Nous retiendrons la méthode la plus simple. Elle consiste à définir la zone de recherche comme les M documents négatifs les plus proches du barycentre des exemples positifs (on notera alors $QZ_M(c_j)$ l'ensemble de ces exemples).

$$R_{jk} = \max \left\{ \begin{array}{l} \frac{t}{|c_j|} \sum_{d_i \in c_j} d_{ik} - \frac{1-t}{|QZ_M(c_j)|} \sum_{d_i \in QZ_M(c_j)} d_{ik} \\ 0 \end{array} \right. \tag{4.2}$$

L'utilisation des élements négatifs et du Query Zoning permettent d'obtenir un profil plus précis pour la classe, ce qui est précisément le problème du filtrage. Nous pensons que cette technique est beaucoup moins utile dans le cas du routage. En effet, dans ce modèle de tâche, le profil n'a pas besoin d'être correct, il suffit que le profil de la bonne classe soit meilleur que les autres. Un terme peu discriminant et présent dans tout le corpus peut être conservé dans chaque profil. Son poids sera sensiblement le même pour chaque classe et son importance dans la prise de décision sera quasiment nulle (puisque la décision revient à comparer les valeurs de confiance de chaque classe). C'est pourquoi, nous pensons qu'en routage, les éléments négatifs sont peu utiles. Il est tout à fait possible de prendre $t = 1$ (le Query Zoning n'est a fortiori pas utilisé non plus).

Pour valider cette analyse, nous avons comparé l'utilisation du Query Zoning dans un environnement de filtrage et dans un environnement de routage. Comme dans la section précédente, nous avons utilisé le corpus 20 Newsgroups. Dans un premier temps, nous avons appris un profil par classe avec la formule de Rocchio soit sans les éléments négatifs ($t = 1$), soit avec tous les éléments négatifs ($t = 0.8$) soit avec les 1000 éléments négatifs les plus proches de chaque classe (Query Zoning avec $t = 0.8$). Nous avons ensuite mesuré les valeurs de similarité entre chaque document de test et chaque catégorie. À partir de ces valeurs, nous avons mesuré deux taux de performance dans une optique de filtrage

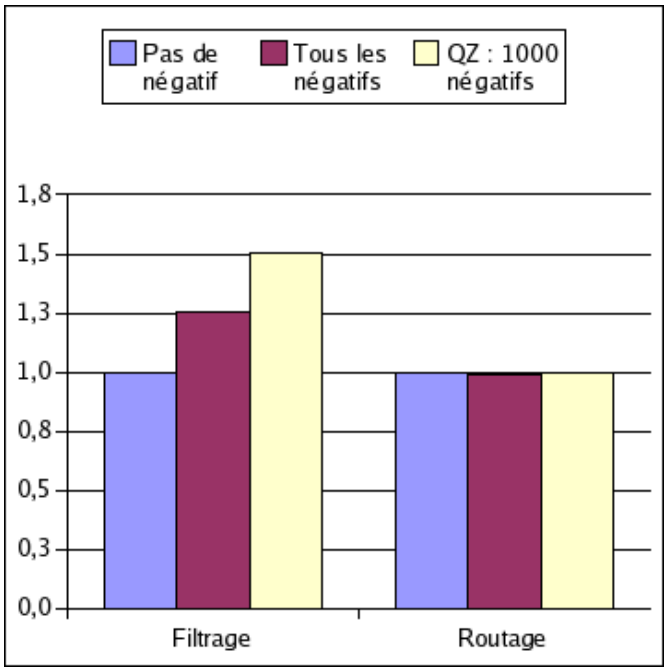


FIG. 4.2 – Performance en filtrage et en routage de Rocchio avec ou sans Query Zoning. Les valeurs ont été normalisées afin que la performance de Rocchio sans élément négatif ($t = 1$) soit 1.

ou de routage. En filtrage, la mesure F_1 (voir section 2.6.1) est calculée pour toutes les valeurs de seuils possibles, et le meilleur taux est conservé. En routage, nous avons utilisé le taux de bonne classification. Les résultats sont présentés dans la figure 4.2. Ils sont proches de ceux de [SMB97] : nous obtenons pour le filtrage une amélioration de 20% avec le Query Zoning par rapport à Rocchio avec éléments négatifs, tandis que Singhal obtient entre 5% et 15%. Il y a également une amélioration de 30% entre Rocchio avec et sans élements négatifs (A. Singhal ne présente pas de résultat de comparaison sur ce point). Contrairement au filtrage, les élements négatifs et le Query Zoning ne sont d'aucune utilité en ce qui concerne le routage. Ceci conforte notre analyse. Cette différence de comportement est d'autant plus significative que les valeurs de similarité entre classes et documents sont les mêmes dans les deux cas et que seules les mesures de performance ont changé.

4.4 Systèmes automatiques ou semi-automatiques

4.4.1 Étude des deux modèles d'interaction

Jusqu'à présent, les systèmes de CT ont toujours été conçus de façon à fonctionner de manière complètement autonome. Ils doivent permettre de remplacer un expert humain

en effectuant le classement automatiquement. Actuellement, les besoins en CT sont très importants. Des systèmes sont mis en place alors que la technologie n'est pas toujours suffisamment fiable et que le classifieur fait encore des erreurs. Lorsque l'application ne peut pas tolérer d'erreur sur la classification, il est préférable de conserver un opérateur humain pour valider et éventuellement modifier les propositions de la machine. Cette approche permet, malgré tout, de diminuer les temps de traitement par rapport à un fonctionnement complètement manuel. Le système est alors dit "semi-automatique".

Dans le cas du filtrage, le système semi-automatique est réglé de façon à conserver tous les documents potentiellement intéressants. L'opérateur humain doit ensuite relire ces documents pour sélectionner ceux qui portent réellement sur la thématique recherchée. Ce principe permet de soulager le travail de l'opérateur puisque le premier filtre effectué par la machine permet déjà de supprimer un nombre important de documents.

Dans le cas du routage, un système semi-automatique peut être utile lorsque le nombre de classes n est très grand. Lorsque tout est manuel, l'opérateur doit sélectionner une classe parmi n. Il doit donc parcourir toute la liste pour déterminer la meilleure catégorie. Avec un système semi-automatique, l'algorithme sélectionne les x classes les plus probables (avec $x \ll n$) et les présente à l'opérateur. Ce dernier n'a plus qu'à lire les x propositions et choisir l'une d'entre elles. Pour que cela permette effectivement un gain de temps, il faut que la bonne réponse soit toujours parmi les x premières propositions. En effet, dans le cas contraire, l'opérateur doit d'abord lire les x propositions, réaliser que aucune n'est correcte puis faire un choix manuel parmi les $n - x$ classes de départ. Le routage semi-automatique a déjà été exploité pour la classification de courriers électroniques [SK99a]. Dans cet article, les auteurs montrent que l'utilisation de trois propositions ($x = 3$) au lieu d'une permet de rendre le classifieur plus fiable sans gêne supplémentaire pour l'utilisateur par rapport à la situation où seule la première proposition est affichée.

Les mesures de performance utilisées pour comparer les différents algorithmes doivent refléter le degré de satisfaction des utilisateurs. Dans les deux cas (automatique ou semi-automatique), la satisfaction ne s'exprime pas de la même façon. Dans le contexte automatique seule la première proposition est importante, alors que dans le contexte semi-automatique il suffit que la bonne classe soit trouvée parmi les x premières propositions pour que l'on juge que le document est correctement classé. Ceci nous amène à définir une nouvelle mesure de performance : perf-x. Perf-x est égal à 1 si la classe correcte se trouve parmi les x premières propositions, 0 dans le cas contraire. Il n'y a pas de pénalité à trouver la classe correcte en deuxième position plutôt qu'en première. Comme perf-1 est exactement identique au taux de bonne classification, perf-x est une simple extension de la mesure standard.

Comparer deux algorithmes avec perf-x revient à comparer les algorithmes sur les

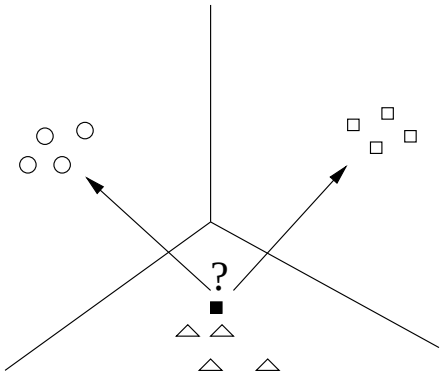
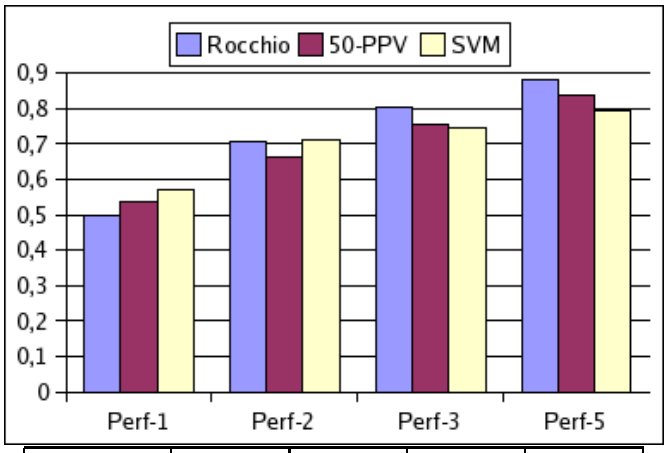


FIG. 4.3 – Illustration de l'utilité de la mesure perf-2.

Étant donné la position de l'exemple noir à classer, tous les catégoriseurs vont l'attribuer à la classe triangle. Pourtant, ce n'est pas sa catégorie. Le classifieur doit donc fournir une deuxième classe possible, rond ou carré. Les algorithmes discriminants sont démunis pour répondre à ce genre de questions car dans cette zone les frontières pertinentes ne s'occupent que de séparer les triangles des autres classes, pas de discriminer les ronds des carrés.

exemples *bruités*. Les exemples bruités sont des documents qui appartiennent à une catégorie mais dont le contenu correspond naturellement plutôt à une autre. En effet, lorsqu'un document est mal classé par la première proposition, c'est ou bien parce que le classifieur s'est trompé, ou bien parce que l'exemple est bruité et correspond effectivement plus à la classe proposée qu'à sa classe correcte. Dans le premier cas, un algorithme plus performant pourrait trouver la classe correcte en premier. La mesure perf-1 permet alors de départager deux algorithmes (le meilleure étant celui qui trouve la classe correcte en première proposition). Dans le second cas, la mesure perf-1 ne permet pas de comparer deux algorithmes car aucun ne peut trouver la classe correcte en première proposition (selon perf-1, la classification est donc incorrecte quel que soit l'algorithme). La comparaison ne peut s'effectuer que sur les choix suivants (par exemple un algorithme qui trouve la bonne classe en deuxième choix est plus performant qu'un algorithme qui la trouve en troisième choix). Il faut donc utiliser les mesures perf-x. La figure 4.3 illustre la mesure perf-2 par un exemple. Lorsqu'un document bruité de classe « carré » apparaît au centre de la classe « triangle », tous les algorithmes ont tendance à proposer la classe « triangle » pour ce document. Pour savoir si un algorithme est plus performant à classer ce document, il faut utiliser la mesure perf-2 (qui permet de mesurer si la classe correcte « carré » est trouvée en deuxième choix ou non).

Cette mesure a déjà été utilisée pour la classification de courriers électroniques [BSA00, SK99a] mais les auteurs respectifs n'ont pas étudié l'effet induit par cette nouvelle mesure sur les algorithmes. L. Larkey et W. Croft [LC96] comparent k-PPV, Rocchio et



	perf-1	perf-2	perf-3	perf-5
Rocchio	0.50	0.71	0.8	0.88
50-PPV	0.54	0.67	0.76	0.84
SVM	0.57	0.71	0.74	0.79

FIG. 4.4 – Comparaison des performances relatives de Rocchio, k-PPV et SVM suivant la mesure perf-x avec x variant de 1 à 5.

Naïve Bayes sur perf-1 et perf-10 : k-PPV est meilleur que Rocchio et Naïve Bayes sur perf-1 mais moins performant sur perf-10. Ils concluent que k-PPV a une très bonne précision à un faible taux de rappel mais qu'il est moins compétitif pour de forts taux de rappel.

4.4.2 Expérimentations avec perf-x

Nous avons réalisé une expérience supplémentaire pour mettre en évidence les modifications induites par l'utilisation de ces nouvelles mesures de performance. Nous n'avons pas pu utiliser le corpus 20 Newsgroups comme dans les deux expériences précédentes, parce que les performances sont de 100% quel que soit l'algorithme utilisé dès que x est supérieur ou égal à 2. Elles ne permettent donc pas de comparer les systèmes. Nous avons utilisé le corpus Mail Center (présenté à la section 3.4). Les résultats sont donnés dans la figure 4.4.

Plus x augmente plus les performances perf-x sont élevées. En effet, si un document a été correctement classé avec perf-x, il sera forcément bien classé avec perf-$(x+1)$ et donc perf-$(x+1)$ est forcément plus élevé que perf-x. Nous nous intéressons ici aux performances relatives de chaque algorithme par rapport aux autres. Pour perf-1, la hiérarchie est habituelle (voir [Yan99]) : Rocchio est inférieur à k-PPV, lui-même inférieur à SVM. Plus x augmente, plus la hiérarchie s'inverse pour atteindre à perf-5 : SVM inférieur à k-PPV lui-même inférieur à Rocchio. Ce résultat est similaire à celui présenté par Larkey et Croft [LC96].

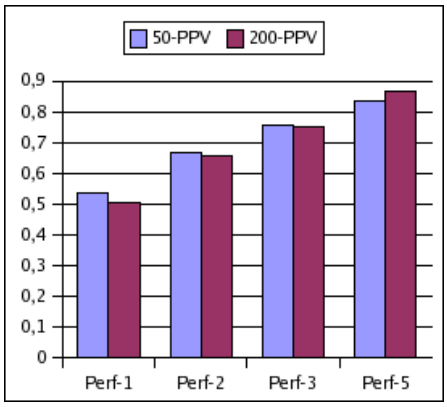
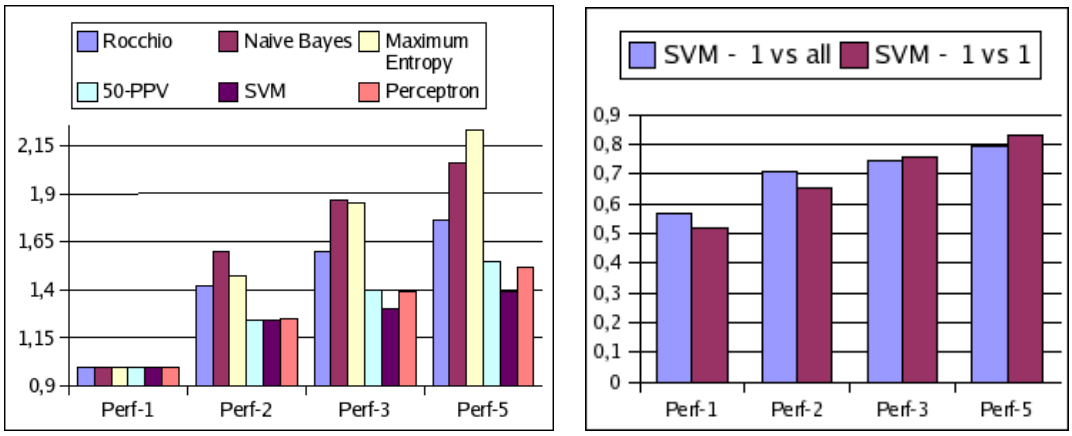


FIG. 4.5 – Expériences supplémentaires avec k-PPV

Rocchio est peu performant avec perf-1, mais il devient clairement dominant avec perf-x pour x supérieur à 2. Nous avons montré dans [VY02] que Rocchio est très résistant au bruit et que le corpus Mail Center est très bruité. Comme les mesures perf-x permettent de départager les algorithmes sur les exemples bruités, Rocchio est très performant pour cette mesure.

Quant à k-PPV, le taux de performance dépend beaucoup de la valeur de k. Il est clair que plus x augmente, plus il faut élargir la zone des voisins (donc augmenter k) pour avoir des valeurs fiables sur les x premières classes. La valeur optimale de k est différente suivant la mesure perf-x utilisée. Pour vérifier ceci, nous avons également mesuré les taux de performance avec $k = 200$. La figure 4.5 montre ce comportement : 200-PPV est moins performant que 50-PPV pour $x \leq 3$ et plus performant pour $x > 3$. Nous avons également réalisé des tests avec des valeurs supérieures à 200 mais les performances étaient inférieures pour toutes les mesures. Au-delà d'une certaine valeur, le principe de localité qui régit l'algorithme ne se vérifie plus (des exemples trop éloignés sont pris en compte pour le vote) et les performances chutent. Cela signifie donc que la valeur de k doit être optimisée pour chaque mesure perf-x. Enfin, dans tous les cas de figure, k-PPV ne parvient jamais à faire mieux que Rocchio pour perf-5 (comme la valeur de k est limitée par la taille du corpus, cela reste à valider avec un corpus plus important).

Le cas de SVM est encore plus impressionnant : il est très performant sur perf-1 et médiocre sur perf-5. Cela signifie que SVM trouve plus souvent la bonne classe que les autres algorithmes, mais lorsqu'il se trompe, les autres suggestions sont très souvent fausses également. Ceci peut s'expliquer par le caractère discriminant de l'algorithme. Dans les algorithmes discriminants, les frontières sont construites par mise en concurrence des classes entre elles. Lorsqu'un document bruité apparaît dans une zone qui ne correspond pas à sa classe, les frontière calculées ne sont pas pertinentes et les classes proposées sont erronées. Dans les algorithmes génératifs, les classes ne sont pas en concurrence. Si un document est dans une zone plus proche d'une autre classe que de sa



Algorithmes discriminants et génératifs SVM multi-classe

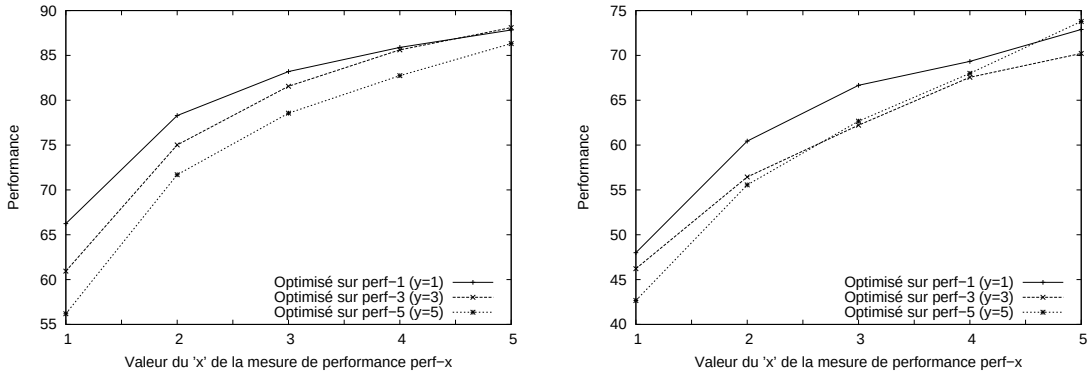
FIG. 4.6 – Expériences supplémentaires avec SVM.

classe correcte, cela n'empêche pas cette dernière d'avoir un taux de confiance à peu près fiable. Pour vérifier cette hypothèse, nous avons comparé les performances avec les algorithmes génératifs Naïve Bayes et Maximum d'Entropie [NLM99] (en utilisant le logiciel Bow de McCallum [McC96]) et un autre algorithme discriminant : un réseau de neurones Perceptron (en utilisant l'implémentation du logiciel Snow de Dan Roth [CCJ+99]). Les résultats présentés dans la partie gauche de la figure 4.6 montrent perf-x pour chaque algorithme relativement à sa valeur de perf-1 (contrairement au graphique précédent où les valeurs étaient absolues). Les trois premiers algorithmes (Rocchio, Naïve Bayes et Maximum d'Entropie) sont génératifs alors que les trois suivants (k-PPV, SVM et Perceptron) sont discriminants. La différence de comportement entre les deux approches est visible : les algorithmes génératifs s'améliorent plus que les algorithmes discriminants lorsque x augmente.

Une autre caractéristique de SVM qui peut expliquer sa faible performance est son incapacité à gérer directement des problèmes multi-classes alors que les mesures perf-x avec $x > 1$ sont explicitement construites pour des problèmes multi-classes. Nous avons jusqu'à présent utilisé le modèle « 1 contre tous » (voir section 2.4.2) pour transformer un problème multi-classe en plusieurs sous-problèmes bi-classes. Dans ce cas, lorsqu'un document se trouve dans une zone ne faisant pas partie de sa classe, les frontières ne sont pas pertinentes. C'est le cas par exemple dans notre exemple à trois classes de la figure 4.3. Pour comparer les classes *rond* et *carré*, il faut en fait comparer *rond* contre *carré+triangle* et *carré* contre *rond+triangle*. Dans la zone *triangle*, les frontières pour comparer *rond* et *carré* sont surtout dépendantes de la classe *triangle* et donc inefficaces pour comparer les deux autres classes. Lorsque le modèle est « 1 contre 1 », la comparaison *rond* contre *carré* est effectuée sans intervention des documents de la zone *triangle*. Les frontières sont donc correctement dessinées même dans cette

zone et les mesures perf-x avec $x > 1$ doivent être meilleures qu'avec le modèle « 1 contre tous ». La partie droite de la figure 4.6 présente les résultats des mesures perf-x pour SVM en mode « 1 contre tous » ou « 1 contre 1 ». Ils confirment notre analyse puisque « 1 contre 1 » est bien plus performant que « 1 contre tous » lorsque $x > 1$. En revanche, « 1 contre 1 » est bien plus mauvais pour perf-1 (du même ordre que Rocchio). Comme pour k-PPV, le système ne peut pas être optimisé pour toutes les mesures perf-x simultanément.

Ces résultats peuvent également s'expliquer par le fait que les algorithmes discriminants cherchent souvent explicitement à minimiser le taux d'erreur (c'est-à-dire maximiser perf-1). Ils ont donc des performances supérieures sur perf-1 que sur d'autres mesures. Dans le cas où nous sommes intéressés par perf-x, il faut alors prendre des algorithmes qui optimisent le classifieur sur cette mesure perf-x. S'il est difficile de modifier SVM pour lui faire prendre en compte ce type de mesure, c'est relativement aisé avec les algorithmes guidés par les erreurs comme le Perceptron. Dans ce type d'algorithme, les paramètres du classifieur ne sont modifiés que lorsqu'un exemple est mal classé. Pour l'optimiser sur perf-x, nous lui permettons de modifier ces paramètres uniquement si l'exemple est mal classé suivant perf-x, c'est-à-dire si la classe correcte de l'exemple ne se trouve pas parmi les x classes les plus pertinentes (et non plus seulement la première classe). Pour voir l'effet de cette optimisation sur les performances, nous avons réalisé une expérience en mesurant la valeur perf-x après avoir optimisé le classifieur pour une mesure perf-y (y non nécessairement égal à x). Cette expérience a été réalisée sur plusieurs corpus du répertoire UCI [BM98b] (Covtype, Digits, Isolet et Letter-recognition) ainsi que les corpus Newsgroups et Mail Center (voir section 3.4). La figure 4.7 illustre les résultats sur les deux corpus textuels (les résultats sont similaires pour les autres corpus).



Newsgroups Mail Center

FIG. 4.7 – Performance du Perceptron suivant la mesure pour laquelle il a été optimisé.

Si l'optimisation lors de l'apprentissage était réellement efficace, pour toutes les valeurs de x, la mesure perf-x devrait être maximale lorsque l'apprentissage a été optimisé pour perf-y avec $y = x$. Or, la valeur optimale semble être généralement obtenu pour y légèrement inférieur à x. Par exemple, mesuré avec perf-3, le classifieur est plus performant s'il a été optimisé pour perf-1 que pour perf-3. Au moins pour $x < 5$, l'apprentissage standard (optimisé sur perf-1) semble obtenir des performances au moins aussi bonne qu'avec toute autre optimisation. Il est possible dans certains cas d'obtenir de meilleures performances que l'apprentissage standard mais il semble difficile de trouver la meilleure optimisation et la valeur la plus logique ($y = x$) n'est pas la meilleure solution.

Au vu de tous ces tests, nous pouvons affirmer que les algorithmes discriminants sont généralement peu performants en mode semi-automatique. Rocchio qui semble à l'heure actuelle le plus performant des algorithmes génératifs est donc le mieux adapté à ce mode de fonctionnement.

4.5 Influence du niveau de bruit des corpus

4.5.1 Définition du bruit

Nous avons vu dans le chapitre précédent que les corpus des nouvelles applications sont souvent constitués avec moins de soin que les corpus de référence. Ils sont donc souvent bruités. Le terme de *bruit* est défini dans la théorie du signal comme un élément perturbateur aléatoire qui vient brouiller le signal contenant de l'information. Ce terme a été repris dans le cadre de la catégorisation pour décrire les éléments aléatoires qui viennent géner la classification. Ce bruit peut être contenu dans les documents eux-mêmes, sous forme de fautes d'orthographe ou de syntaxe. Tous les termes non discriminants peuvent également être considérés comme du bruit. Il peut également être présent dans les assignations d'étiquettes de classe incohérentes ou erronées lorsqu'un document qui appartient à une classe a été attribué par erreur à une autre classe lors de la constitution du corpus. La figure 4.8 illustre les deux types de bruit.

Le premier type de bruit est très fréquent dans le domaine textuel. Un document est typiquement représenté par plusieurs dizaines de milliers d'attributs et on sait par avance qu'une infime fraction d'entre eux seront utiles pour la catégorisation. Tout classifieur doit donc être capable de prendre en compte ce type de bruit. Le deuxième type est en revanche plus rare. Tous les corpus standards sur lesquels les algorithmes sont comparés sont généralement constitués avec beaucoup de soin afin d'offrir des corpus de référence de qualité. Dans le cas d'applications réelles, nous pensons que ce type de bruit est beaucoup plus fréquent. C'est le cas dans le Mail Center où les opérateurs devant traiter les mails le plus rapidement possible peuvent fournir une mauvaise réponse, dans

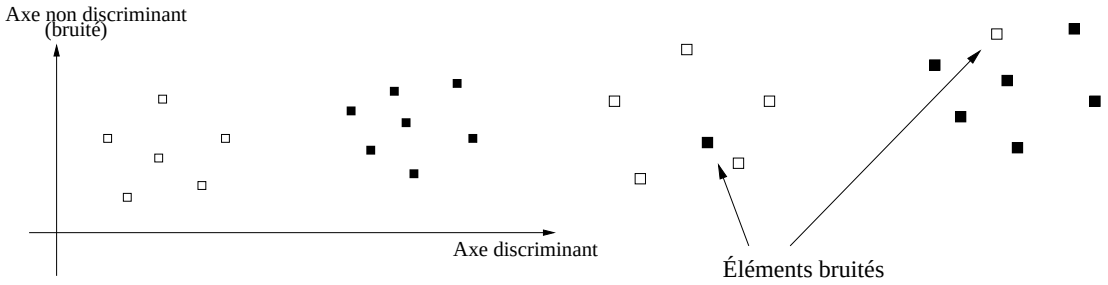


FIG. 4.8 – Illustration des types de bruit : dans le premier cas, il s'agit d'un bruit sur les documents (un attribut est non pertinent pour la classification) ; dans le deuxième cas, c'est l'assignation des étiquettes de classes qui est bruitée.

la classification des mails personnels ou même dans le cas du Spam où la limite entre mails non sollicités et mails légitimes est parfois floue et variable suivant les utilisateurs.

4.5.2 Expérimentations sur corpus artificiellement bruités

Il est donc important d'étudier le comportement des divers algorithmes en fonction du second type de bruit. Afin de visualiser réellemment les effets dûs au bruit, nous avons réalisé des expériences sur des corpus artificiellement bruités. Ces corpus sont construits en changeant aléatoirement l'étiquette de $x\%$ des exemples (pour $x = 0, 10, 20..., 90$). Nous avons effectué cette expérience sur le corpus 20 Newsgroups avec Rochio, k-PPV et SVM. Les résultats sont synthétisés dans la figure 4.9.

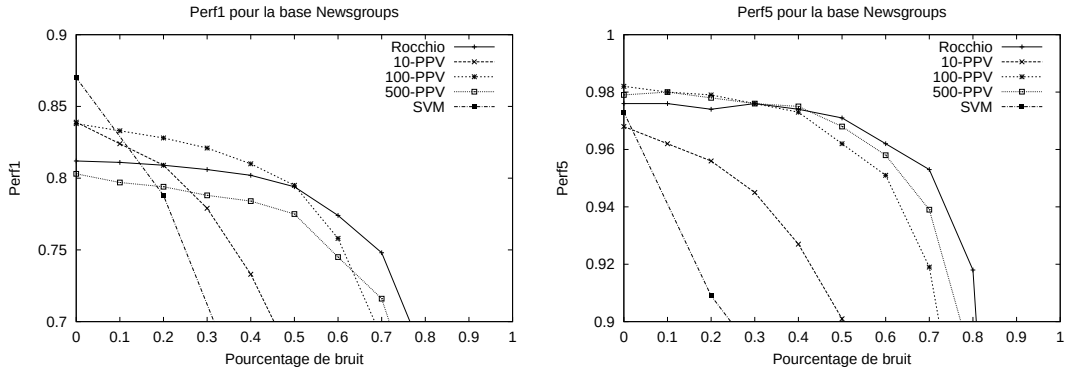


FIG. 4.9 – Performance suivant la quantité de bruit.

Ces graphiques indiquent clairement que Rocchio est l'algorithme le moins sensible au bruit. Ses performances restent quasiment inchangées avec près de 50% de documents qui ont une étiquette de classe aléatoire ! SVM, en revanche, qui s'appuie sur les éléments aux frontières est très sensible au bruit et ses performances chutent énormément dès que le bruit apparaît. Il faut néanmoins nuancer ce résultat par le fait qu'il existe un

paramètre dans SVM permettant de prendre en compte la quantité de bruit présent dans un corpus. Nous n'avons pas fait d'expérience en jouant sur ce paramètre. Nous l'avons toujours laissé à sa valeur par défaut qui convient pour tous les corpus de textes standards mais risque de s'avérer insuffisant dans le cas de nos expériences où les corpus sont fortement modifiés. Le comportement de k-PPV dépend essentiellement de la valeur de k. L'algorithme est de moins en moins sensible au bruit au fur et à mesure que k augmente. Au-delà de $k = 100$, les performances restent stables lorsque le bruit augmente mais les performances sont faibles lorsque le bruit est faible. La valeur de k est donc comme toujours un compromis entre la sensibilité au bruit et la conservation de l'aspect local de l'algorithme (en l'absence de bruit, la meilleure valeur de k est à l'évidence 1).

4.6 Conclusion

Nous avons étudié dans ce chapitre, à travers quatre exemples, que le contexte applicatif est très important pour comparer les performances des algorithmes.

La racinisation est habituellement considérée comme plutôt efficace en RI. Elle est également souvent employée en CT parce qu'elle permet de raccourcir les temps de calcul. Or, nous avons montré qu'elle peut avoir un effet désastreux sur les performances en CT et qu'elle ne devrait pas être utilisée dans ce contexte.

Le Query Zoning est très efficace en filtrage. Il est aujourd'hui considéré comme un moyen simple pour améliorer Rocchio. Pourtant, nous avons montré qu'il n'améliore pas les performances en routage et qu'il est donc complètement inutile dans une application avec ce mode de classification.

Jusqu'à présent, toutes les mesures de performance qui sont utilisées considèrent que les applications utilisent la CT en mode automatique. Dans ce cadre, seule la première proposition de l'algorithme est utilisée et le fait que la réponse correcte soit en seconde position ou à la dernière importe peu. Ce n'est pourtant pas le cas dans de nombreuses applications qui sont mises en place actuellement où les classifieurs n'ont qu'un rôle de conseiller. Lorsque les mesures sont généralisées pour refléter un mode d'utilisation plus proche de la réalité, les performances des classifieurs ne sont plus du tout semblables. SVM semble à l'heure actuelle l'algorithme le plus performant lorsque la mesure est classique. En revanche, lorsque la mesure prend en compte l'aspect semi-automatique, SVM obtient des résultats inférieurs aux autres algorithmes et Rocchio devient le plus performant.

Par ailleurs, les corpus de référence sur lesquels sont réalisées les expérimentations sont généralement de meilleure qualité que les corpus utilisés dans les nouvelles applications. Dans le cas de corpus très bruité (avec beaucoup de documents dont l'étiquette de classe est incorrecte), Rocchio semble très robuste alors que SVM est au contraire

très sensible aux erreurs d'étiquetage.

Les nouvelles applications de la CT que nous avons présentées au chapitre trois cumulent souvent ces deux dernières caractéristiques. Dans les deux cas, elles tendent à défavoriser les algorithmes SVM ou k-PPV. Rocchio peut alors s'avérer le plus performant. Pour cette raison et pour bénéficier de son efficacité computationnelle, nous avons décidé d'étudier plus en détail son comportement dans la suite. Nous nous penchons dans le chapitre suivant sur le problème de la CT dans un contexte où les catégories sont non-thématiques.

Chapitre 5

Rocchio Multi-Prototypes

Sommaire du chapitre

5.1 Introduction

Dans le chapitre 4, nous avons étudié les conséquences de quelques spécificités des nouvelles applications dont nous avons présenté quatre exemples au chapitre 3. Nous avons ainsi pu constater les différences entre filtrage et routage et montrer que les algorithmes optimisés pour l'une des approches ne sont pas nécessairement performants dans l'autre. Nous avons également mis en évidence que les classifieurs reposant sur

l'apprentissage discriminant sont souvent moins robustes au bruit et moins adaptés à un mode d'utilisation semi-automatique que les algorithmes génératifs. Toutes ces analyses semblent indiquer que, pour les nouvelles applications, les algorithmes génératifs peuvent être plus performants que les algorithmes discriminants. C'est par exemple le cas pour le problème du Mail Center où Rocchio obtient 88% de performance avec perf-5 alors que SVM, pourtant réputé meilleur, obtient seulement 79%.

Il nous reste à étudier une dernière caractéristique importante qui intervient dans beaucoup de nouvelles applications opérationnelles : le fait que les catégories ne sont pas définies par une thématique mais par un groupement de plusieurs thèmes. Contrairement aux caractéristiques étudiées précédemment, cette particularité ne semble pas, a priori, gênante pour les algorithmes discriminants. En effet, ces derniers ne s'intéressent qu'aux frontières entre classe. Nous savons que, dans le domaine textuel, en raison de l'espace de recherche de très grande dimension, les classes sont généralement linéairement séparables. Que les classes soient complexes ou non (constituées d'un ou plusieurs thèmes), la surface de séparation entre classes reste en général linéaire. Les algorithmes discriminants, puisque la complexité des surfaces de séparation ne change pas, doivent parvenir à optimiser la classification aussi bien que pour des classes plus simples. En revanche, les algorithmes génératifs cherchent précisément à modéliser le contenu des classes. Ils doivent donc impérativement prendre en compte la multi-thématicité des catégories. Si les modèles gérés par les classifieurs sont suffisamment complexes pour prendre en compte ce type de classes (par exemple les modèles à base de mélanges de gaussiennes), ils ne seront pas affectés. Les algorithmes qui modélisent les classes par un thème central (comme Rocchio ou Naïve Bayes) doivent, en revanche, avoir un moyen pour gérer la complexité de ces classes sous peine de voir leur performance chuter dans le cas de catégories multi-thématiques. Cette idée a été émise parallèlement à nos travaux lors de la conférence ECML 2003 par R. Vilalta et I. Rish [VR03]. Les auteurs suggèrent d'utiliser le clustering pour améliorer Naïve Bayes mais leur solution est actuellement moins aboutie que celle que nous proposons ici.

Nous présentons dans ce chapitre une extension de l'algorithme Rocchio, appelée RMP pour Rocchio Multi-Prototypes, qui modélise une catégorie par un ensemble de sous-classes thématiquement homogènes. Dans la section 5.2, nous décrivons plusieurs algorithmes existants qui utilisent la notion de sous-classes. Dans la section 5.3, nous présentons une expérience préliminaire pour mettre en évidence les hypothèses qui guideront la construction de notre algorithme. RMP est décrit en détail dans la section 5.4. Enfin, la section 5.5 décrit les expériences réalisées, les analyses du comportement de RMP et les comparaisons avec k-PPV et SVM.

5.2 État de l'art : détection de sous-classes

5.2.1 Clustering

Tous les algorithmes de catégorisation qui utilisent des sous-classes emploient le clustering (ou apprentissage non supervisée) pour les détecter. Le clustering est un modèle de tâche dans lequel l'algorithme partitionne un ensemble de données de façon à regrouper les textes proches (au sens d'une mesure de similarité à définir ; dans le domaine du texte, nous utilisons les mesures de similarité standard définies au chapitre 2). Il existe essentiellement trois familles : les algorithmes hiérarchiques, incrémentaux ou séquentiels. Si le clustering hiérarchique est réputé comme le plus performant, c'est aussi le plus coûteux en temps de calcul puisque la complexité algorithmique est en $O(N^2)$ alors que les deux autres types de clustering sont en $O(N)$ (N étant le nombre de documents du corpus). La littérature est abondante sur le sujet mais aucune technique ne semble faire l'unanimité [SKK00]. Il existe également depuis récemment des techniques de clustering à base d'analyse spectrale [Wei99] mais elles ne sont pas encore utilisées dans le domaine du texte. Nous nous contenterons de présenter les k-Means [HW79], les algorithmes hiérarchiques [Wil88] et le modèle incrémental [WF00].

Pour les k-Means, le nombre K de clusters à construire doit être spécifié préalablement. L'algorithme commence par sélectionner K documents au hasard, qui constituent les profils initiaux. Tous les documents sont ensuite regroupés suivant le profil le plus proche. Les K nouveaux profils sont recalculés comme la moyenne des documents de chaque groupe (le calcul du profil de chaque cluster est donc identique au calcul du profil de chaque classe dans Rocchio sans exemple négatif). Ce processus est itéré jusqu'à ce que plus aucun document ne change de cluster entre deux itérations. La qualité du clustering dépend beaucoup du choix des K centres initiaux. Il est fréquent d'effectuer plusieurs fois l'algorithme afin d'obtenir des résultats plus stables et moins dépendants des centres initiaux.

Dans les algorithmes hiérarchiques, il faut distinguer les approches ascendantes et descendantes. Dans le premier cas, on part d'une partition avec N clusters (un document par cluster). À chaque étape, les deux clusters les plus proches sont fusionnés. Les distances inter-clusters sont recalculées et on recommence jusqu'à n'obtenir plus qu'un seul cluster qui contient tous les documents. Dans le deuxième cas, on part d'un seul ensemble qui contient tous les documents. À chaque étape, un cluster est divisé en deux sous-ensembles et on recommence jusqu'à obtenir un document par cluster. Ces méthodes permettent d'obtenir un ensemble de clusters organisé sous la forme d'un arbre hiérarchique. On obtient une partition des documents en coupant l'arbre à un niveau donné.

Les algorithmes incrémentaux procèdent en incorporant les documents un par un.

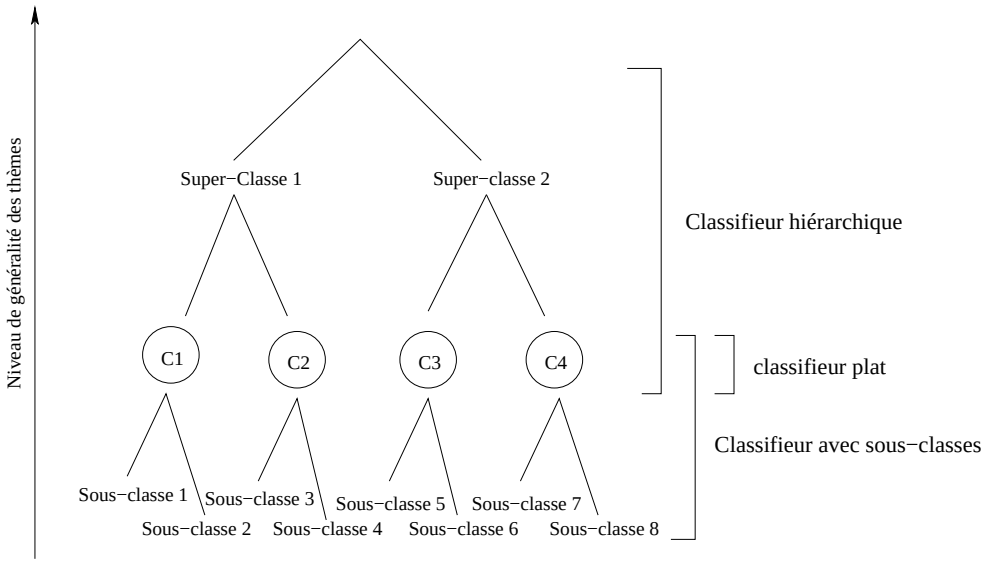


FIG. 5.1 – Opposition entre classifieurs hiérarchiques et classifieurs avec sous-classes. Les catégories à discriminer sont les ensembles C1, C2, ..., C4. Un classifieur "plat" n'utilise que cette information pour faire la classification. Un classifieur hiérarchique utilise des regroupements de classes en ensemble plus généraux alors qu'un classifieur avec sous-classes détermine des sous-ensembles plus précis de chaque classe.

Chaque nouvel exemple est incorporé dans le cluster le plus proche si la distance qui les sépare est suffisamment faible. Si aucun cluster n'est suffisamment proche, le document devient un nouveau cluster à lui tout seul. Cet algorithme permet de construire une partition sans fixer le nombre de clusters a priori et en ne parcourant la liste des documents qu'une seule fois. Malheureusement, les clusters obtenus sont souvent de mauvaise qualité et très dépendants de l'ordre d'arrivée des documents.

5.2.2 Catégorisation avec utilisation de sous-classes

L'idée sous-jacente des classifieurs hiérarchiques (voir section 2.5) est qu'il est plus facile de discriminer les classes du plus général au plus précis plutôt que de chercher directement à détecter la classe correcte. Par exemple, pour reconnaître un document parlant de football, un classifieur hiérarchique va d'abord déterminer que le texte concerne le sport plutôt que l'économie puis différencier les articles qui parlent de football de ceux qui parlent de tennis ; alors qu'un classifieur "plat" standard devra discerner toutes les catégories en même temps aussi bien le football et le tennis que l'agriculture ou la bourse. La hiérarchie sert donc à fournir des catégories plus générales que l'on espère plus faciles à discriminer. La motivation des modèles avec sous-classes est à l'opposée de celle des modèles hiérarchiques (voir la figure 5.1 pour une illustration de cette opposi-

tion). Dans l'utilisation des sous-classes, l'hypothèse est au contraire que les classes sont trop générales et qu'il est préférable de les découper en sous-ensembles plus homogènes, qui seront plus faciles à discriminer les uns par rapport aux autres. Par exemple, il est plus complexe de reconnaître qu'un document parle de sport qui est un thème général sur lequel il est difficile de trouver tous les mots discriminants que de reconnaître qu'un document parle de tennis ou de football pour lesquels des mots très spécifiques existent.

La structuration des classes en sous-ensembles a été utilisée depuis de nombreuses années dans le cadre de l'algorithme des k-PPV pour stocker des corpus de grande taille plus efficacement afin de calculer les similarités inter-documents plus rapidement lors de la classification (comme par exemple les *k-d tree* [FBF77] ou le *Hierarchical Bayesian Clustering* de M. Iwayama [IT95]). Depuis quelques années, quelques travaux tentent d'utiliser les sous-classes pour améliorer la justesse des prédictions.

Le tableau 5.1 résume les différents travaux présentés ci-dessous. Ces travaux utilisent tous le principe suivant (voir la figure 5.2 de la page 90). Un algorithme de clustering commence par déterminer des sous-classes $(\Pi_1, \ldots, \Pi_p)$ à l'intérieur de chaque catégorie $(c_1, \ldots, c_n)$. Un cluster ne peut contenir que des documents provenant de la même classe. Chaque sous-classe est considérée comme une catégorie à part entière. Un algorithme de catégorisation apprend ensuite à discriminer toutes ces sous-classes. Le classifieur est donc une fonction $f : \mathcal{D} \mapsto \{\Pi_1, \ldots, \Pi_p\}$. Pour classer de nouveaux documents dans les classes originelles, les valeurs de confiance pour chaque sous-classe sont calculées et agrégées en une valeur de confiance pour chaque classe par une fonction $g : \{\Pi_1, \ldots, \Pi_p\} \mapsto \{c_1, \ldots, c_n\}$. Cela revient à modéliser chaque classe par un mélange au lieu d'utiliser une distribution unique ou, géométriquement, à essayer de trouver plusieurs surfaces de séparation simples plutôt qu'une seule surface complexe. La fonction d'un tel algorithme est de déterminer la complexité des modèles de chaque classe, l'apprentissage consistant ensuite à apprendre les paramètres du modèle qui aura été fixé. Les différentes techniques diffèrent par :

- l'algorithme de clustering (en particulier le fait qu'il utilise la classe des documents ou pas),
- l'algorithme de classification,
- la méthode d'aggrégation des résultats sur les sous-classes.

Dans [ERS+99, CES00], D. Eichmann décrit le système de filtrage développé à l'université de Iowa pour les conférences TREC [NDa] et TDT [NDb]. Le clustering construit les clusters au fur et à mesure de l'arrivée des documents suivant le processus incrémental. Il est effectué sans information sur les étiquettes de classe (un cluster peut donc contenir des documents de plusieurs classes). Les documents ne sont présentés à l'utilisateur que si le cluster dans lequel ils sont incorporés est de taille suffisante et que la similarité du cluster avec le profil de la classe dépasse un certain seuil (dans le cas

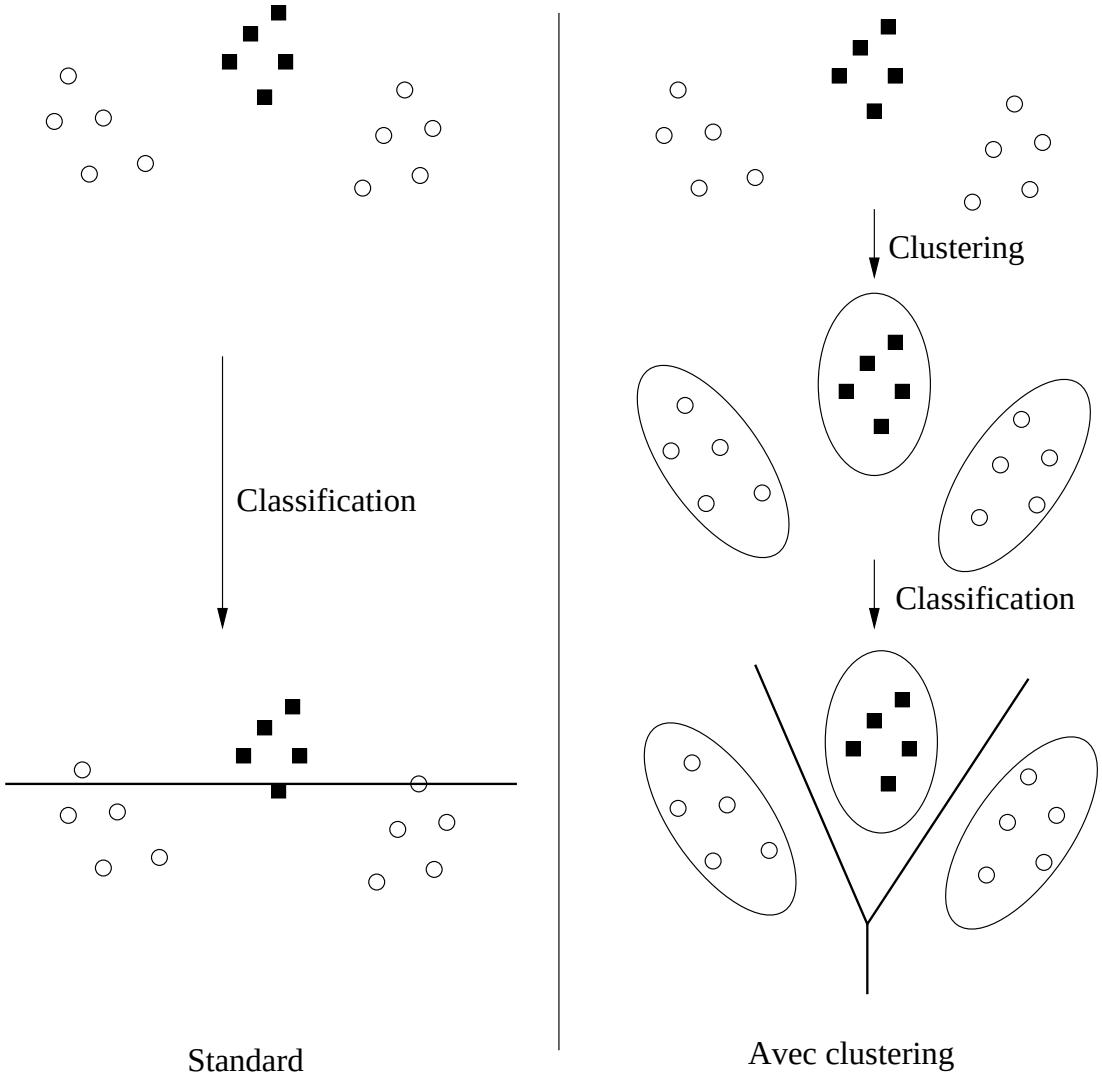


FIG. 5.2 – Principe de la classification avec clustering préalable. Le clustering permet de découper les classes en sous-ensembles plus homogènes qui seront plus faciles à discriminer que la classe de départ.

Article	**Clustering**	**Supervision**	**Classifieur**	**Méthode d'aggrégation**
[ERS+99]	Incrémental	Non	Rocchio	1-PP cluster
[CES00]	Incrémental avec filtrage des clusters non performants	Oui	Rocchio	1-PP cluster
[OLC+01]	Incrémental	Non	SVM	1-PP cluster
[dKMK96]	*Autoclass*	Non	Rocchio	1-PP cluster
[Lam98]	*Generalized Instance Set*	Oui	Rocchio ou Widrow-Hoff	aggrégation linéaire de tous les clusters
[VR03]	*EM*	Non	Naïve Bayes	1-PP cluster

TAB. 5.1 – Caractéristiques des différentes méthodes

La colonne clustering indique le type d'algorithme de clustering utilisé pour construire des sous-classes. La colonne Supervision indique si le clustering utilise des étiquettes des exemples (autrement que pour simplement empêcher deux exemples de classes différentes d'être groupé ensemble). La colonne Représentation des clusters indique comment les documents sont utilisés pour calculer le profil de chaque cluster. La colonne Méthode d'aggrégation indique comment les résultats de classification de chaque cluster sont utilisés pour déterminer la classe des nouveaux documents.

contraire, le document est ajouté au cluster mais n'est pas présenté à l'utilisateur). L'utilisateur peut alors indiquer quels sont les documents pertinents du cluster. Les documents pertinents et non pertinents sont séparés en deux clusters. J.-H. Oh [OLC+01] utilise également le clustering incrémental pour la tâche de filtrage de TREC, en combinaison avec SVM. Dans les deux cas, les résultats sont peu probants (J.-H. Oh obtient même des résultats inférieurs avec le clustering que sans). Ils s'expliquent par l'utilisation d'un algorithme de clustering de mauvaise qualité (clustering incrémental). La qualité est encore dégradée dans le cas de Eichmann car les étiquettes de classe ne sont pas connues (le clustering peut construire des groupes qui contiennent des documents de plusieurs classes en même temps). Dans le cas de Oh, le classifieur utilisé est SVM. Nous avons vu en introduction que la détection des sous-classes est inutile pour les algorithmes discriminants si les surfaces de séparation restent linéaires. D'après le corpus utilisé, il n'y aucune raison pour que ce ne soit pas le cas ici. Enfin, il faut noter que dans les deux cas, l'algorithme de clustering est complètement indépendant de l'algorithme de catégorisation, il ne peut donc pas y avoir de synergie entre les deux.

L'idée de W. Lam [Lam98] est de modifier l'algorithme des k-PPV pour lui donner un comportement plus proche d'un classifieur linéaire (en particulier pour le rendre moins sensible au bruit). L'algorithme de clustering construit les clusters l'un après l'autre. Un premier document est choisi aléatoirement. Les documents les plus proches sont

agrégés au cluster tant que (i) ils sont de même classe et (ii) ils ne sont pas trop loin du barycentre du cluster. Lorsqu'il ne reste plus que des documents d'une autre classe dans la zone proche, le cluster est terminé et un nouveau est commencé. Pour l'apprentissage, un profil par sous-classe est appris avec la formule de Rocchio ou de Widrow-Hoff. Un nouveau document est enfin classé par un vote de tous les profils (ce vote permet de conserver une décision linéaire contrairement au cas où seul le profil le plus proche est utilisé). Les résultats présentés indiquent une amélioration significative par rapport à une approche purement k-PPV ou purement linéaire.

Les motivations des auteurs de [dKMK96] sont différentes : il s'agit de structurer un corpus hétérogène avec des documents qui proviennent de plusieurs sources différentes. Certaines sources sont susceptibles de contenir beaucoup d'informations non pertinentes qui peuvent être filtrées préalablement par une utilisation judicieuse de sous-classes. Le partitionnement est effectué par l'algorithme *autoclass* [CKS+88]. Une phase de validation permet à un Perceptron d'attribuer à chaque cluster un poids qui représente l'intérêt potentiel du cluster. Ceci permet de limiter l'influence des documents bruités en les regroupant dans des clusters puis en leur donnant un poids faible pendant la phase de validation avec le Perceptron. Rocchio est ensuite utilisé pour calculer un profil par cluster et la classification est faite par attribution de la classe du prototype le plus proche (pondéré par le poids du cluster). Les résultats indiquent que cette procédure est un moyen efficace d'isoler les clusters les moins utiles et qu'elle permet donc d'améliorer les performances.

Il faut également noter les travaux de Vilalta et Rish [VR03] qui présentent une analyse formel du comportement de Naïve Bayes lorsque le concept à décrire est constitué de deux formes disjointes. Ils indiquent que l'utilisation du clustering permet de résoudre ce problème et montrent expérimentalement que c'est effectivement le cas, mais ils ne proposent pas de méthode spécifique pour effectuer ce clustering. Ces travaux ont été publiés au même moment que notre propre système de clustering.

Enfin, on peut également évoquer les travaux de Nigam et McCallum [NMTM00] qui utilisent des modèles de mixture pour représenter les classes négatives (dans le cadre du filtrage).

Les deux travaux qui obtiennent des résultats concluants sont ceux qui utilisent des données de supervision (directement dans l'algorithme de clustering pour Lam et dans le calcul des pondérations par Perceptron pour Kroon). Nous pensons donc que l'utilisation des étiquettes de classe est une information importante à prendre en compte. Par ailleurs, à l'exception de Vilalta et Rish, aucun de ces travaux ne cherche explicitement à modéliser des catégories multi-thématiques. Nous souhaitons mettre au point un algorithme de clustering qui ne construit des sous-classes que dans le cas où les catégories contiennent effectivement plusieurs thèmes différents.

G1. comp.graphics	G11. alt.atheism
G2. comp.windows.x	G12. sci.electronics
G3. comp.os.ms-windows.misc	G13. sci.crypt
G4. comp.sys.mac.hardware	G14. sci.space
G5. comp.sys.ibm.pc.hardware	G15. sci.med
G6. talk.politics.guns	G16. misc.forsale
G7. talk.politics.mideast	G17. rec.sport.baseball
G8. talk.politics.misc	G18. rec.sport.hockey
G9. talk.religion.misc	G19. rec.autos
G10. soc.religion.christian	G20. rec.motorcycles

Base 1 : Classes thématiques	**Base 2** : Classes non thématiques
C1 : G1,G2,G3,G4,G5	C1 : G1,G2,G6,G12,G16
C2 : G6,G7,G8,G9,G10,G11	C2 : G3,G7,G8,G13,G17
C3 : G12,G13,G14,G15	C3 : G4,G9,G10,G14,G18
C4 : G16,G17,G18,G19,G20	C4 : G5,G11,G15,G19,G20

TAB. 5.2 – Liste des groupes de discussion pour le corpus 20 Newsgroups.

5.3 Expérience préliminaire

Avant de développer un algorithme de clustering spécifique à notre problème, il faut nous assurer que l'utilisation de sous-classes peut effectivement améliorer les performances du classifieur (autrement dit vérifier que l'hypothèse indiquant que des sous-classes homogènes sont plus faciles à discriminer que des classes complexes plus générales). Pour vérifier cette hypothèse, nous avons réalisé une expérience en considérant des sous-classes optimales (qui correspondent réellement à des sous-thèmes) plutôt que le résultat d'un clustering (auquel cas les variations de performance pourraient être dûes à un mauvais clustering plutôt qu'à l'utilisation de sous-classes proprement dit). Nous avons une nouvelle fois utilisé le corpus 20 Newsgroups dans lequel les groupes de discussion sont classés dans une hiérarchie thématique propre à Usenet. Les titres des 20 groupes, qui reflètent leur position dans la hiérarchie, sont indiqués dans le tableau 5.2. Il est possible de les répartir dans quatre thèmes principaux : `comp` (informatique), `talk` (conversation), `sci` (science) et `rec` (loisir). Trois groupes de discussion ne sont pas dans ces branches de Usenet. Nous les avons arbitrairement placés dans le thème le plus proche, pour avoir des thèmes de taille comparable. `soc.religion.christian` et `alt.atheism` ont été placés dans le thème `talk`, `misc.forsale` a été placé dans `rec`.

À partir de ces quatre thèmes, nous pouvons construire un nouveau corpus (appelé Base 1 dans le tableau 5.2) qui contient non plus vingt mais quatre classes. Chacune des classes est constituée de 5 ou 6 sous-classes qui correspondent aux groupes de discussion d'origine. Les classes sont donc thématiquement homogènes puisque tous les groupes ne constituent qu'un sous-thème du même sujet. Cette base sera appelée « Newsgroups

homogène » pour le reste de ce chapitre. Nous avons également construit un deuxième corpus avec quatre classes que nous appellerons « Newsgroups hétérogène » (Base 2 dans le tableau 5.2). Ces classes sont cette fois constituées d'un ou deux groupes de chacun des thèmes principaux. Les classes construites sont cette fois non-thématiques puisque les sous-classes portent sur des sujets différents. La figure 5.3 illustre les caractéristiques des deux bases.

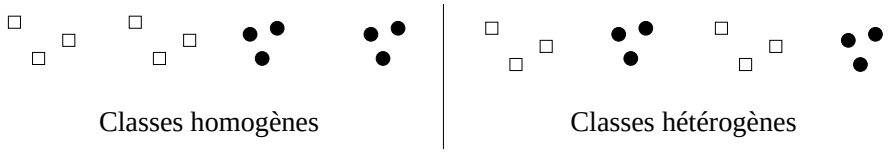


FIG. 5.3 – Exemple de classes homogènes et hétérogènes. Dans les classes hétérogènes les sous-thèmes sont plus proches de sous-thèmes d'autres classes que des sous-thèmes de la même classe.

Dans une première expérience, les quatre classes (C1,..., C4) sont considérées comme des classes à part entière. Les algorithmes apprennent à catégoriser de nouveaux documents dans ces quatre classes. Le classifieur appris est donc une fonction $f : \mathcal{D} \mapsto \{C1, \ldots, C4\}$. Dans une deuxième expérience, les algorithmes apprennent à discriminer les groupes Usenet. Le classifieur est alors une fonction $f : \mathcal{D} \mapsto \{G1, \ldots, G20\}$. Pour catégoriser un nouveau document dans les classes (C1,...,C4), les classifieurs choisissent la sous-classe (G1,...,G20) la plus pertinente et le document est attribué à la classe qui correspond à cette sous-classe (d'après le tableau 5.2). C'est le même principe que dans [CES00, OLC+01]. Nous nous intéressons ici à quantifier l'apport des sous-classes par rapport au cas normal sans sous-classe. Les sous-classes étant ici optimales, les résultats que nous obtenons avec ces sous-classes sont les valeurs maximales que l'on peut espérer avec un véritable clustering. Les taux de performance pour chaque expérience sont indiqués dans le tableau 5.3 et la figure 5.4.

	Classes thématiques		Classes non thématiques	
	Sans sous-classes	Avec sous-classes	Sans sous-classes	Avec sous-classes
Rocchio	0.895	0.932	0.750	0.848
30-PPV	0.934	0.940	0.867	0.870
SVM	0.943	0.956	0.874	0.892

TAB. 5.3 – Taux de performance des algorithmes avec ou sans sous-classes

La première observation concerne les différences de comportement entre les trois algorithmes. Rocchio s'améliore significativement avec les sous-classes alors que SVM et k-PPV y sont très peu sensibles. L'analyse présentée en introduction concernant

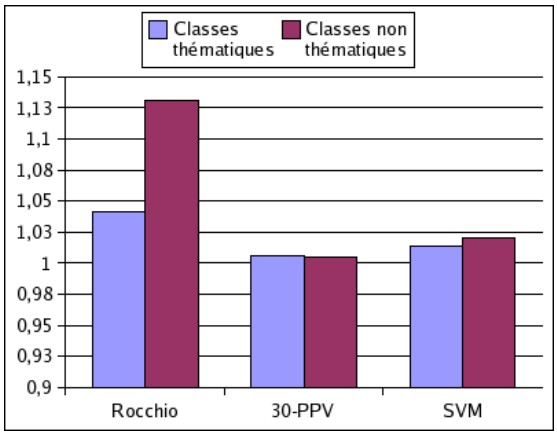


FIG. 5.4 – Taux de performance des algorithmes avec utilisation des 20 sous-classes. Les valeurs correspondent à l'amélioration relative de l'algorithme avec sous-classe par rapport au même algorithme sans sous-classe.

l'inutilité des sous-classes pour les algorithmes discriminants s'avère donc vérifiée. Même dans le cas de classes extrêmement hétérogènes, les surfaces linéaires sont suffisantes pour effectuer une séparation correcte. Rocchio forme aussi un classifieur avec séparation linéaire. Mais comme son apprentissage consiste à modéliser les classes, il doit prendre en compte la forme complexe des classes hétérogènes. L'utilisation de sous-classes apporte dans son cas une amélioration très significative.

Une deuxième remarque est que le bénéfice de l'utilisation des sous-classes n'est pas le même pour les deux corpus. Rocchio ne gagne que 4% de performance sur la première base alors que les sous-classes sont optimales. Dans la première base, lorsque les classes sont thématiquement homogènes, la détection de sous-thèmes n'est pas nécessaire. Ainsi, la catégorie `comp.*` concerne toute l'informatique. Elle est constituée de cinq sous-thèmes plus précis (graphisme, environnement X, système d'exploitation Windows, hardware Mac ou PC). Mais il n'est pas nécessaire de détecter ces sous-thèmes car le thème de l'informatique est déjà suffisamment spécifique pour être reconnu. En revanche, dans la base 2, les classes ne correspondent pas à des thèmes précis. Ce sont juste des ensembles de sous-thèmes sans rapport les uns avec les autres. Ici, l'utilisation de sous-classes permet à Rocchio un gain de 13%. La détection des sous-classes est inutile dans le premier cas mais particulièrement avantageuse dans le second alors que les sous-classes sont les mêmes. Cela montre que l'apport du clustering ne dépend pas uniquement des sous-classes elles-mêmes mais aussi des classes de départ. Le clustering ne sera utile que si les catégories initiales sont thématiquement peu homogènes, c'est-à-dire si certains sous-thèmes sont plus proches de sous-thèmes d'une autre classe que des autres sous-thèmes de la même classe. La motivation pour le clustering se rapproche de celle des travaux de A. D'Alessio [DMKS00]. Ce dernier cherche à modifier les hiérar-

chies en déplaçant des branches d'un emplacement à un autre pour que la hiérarchie soit plus thématique et donc pour qu'elle produise une meilleure catégorisation. Dans le clustering, nous cherchons à subdiviser des catégories qui contiennent des thèmes très différents les uns des autres afin de rendre les clusters plus thématiques.

La détection de classes hétérogènes se fonde sur la recherche des sous-ensembles les plus éloignés les uns des autres. Or, les exemples les plus éloignés sont aussi les plus bruités. Le clustering aura tendance à détecter les groupes de documents bruités qui formeront un cluster constitué uniquement de bruit. L'algorithme doit donc impérativement inclure un mécanisme pour détecter et filtrer ces clusters.

En résumé, pour être utile, un algorithme de clustering doit respecter les contraintes suivantes :

1. Le clustering permet d'améliorer les performances des classifieurs génératifs qui ont un modèle simple. Nous nous intéressons particulièrement à Rocchio.
2. Le clustering ne doit détecter des sous-classes que lorsque les classes ne sont pas thématiquement homogènes et conserver les classes de départ dans le cas contraire. Cela nécessite de prendre en compte dans l'algorithme les étiquettes de classe. Un clustering de ce type sera donc à la fois *faiblement supervisé* et *discriminant* car les étiquettes de classe sont utilisées pour déterminer les clusters à construire.

5.4 Description de RMP

5.4.1 Algorithme général

Soient $\mathcal{C} = \{c_1, ..., c_k\}$ l'ensemble des k classes, et $\Pi = \{\Pi_1, ..., \Pi_p\}$ la partition inconnue des exemples à trouver. $c(x)$ est la classe de l'exemple x et $\Pi(x)$ le cluster de l'exemple x. Nous interdisons la construction de clusters qui mélangent des documents de plusieurs classes. La partition doit donc vérifier :

$$\forall (x, y) \in \mathcal{D},\ \Pi(x) = \Pi(y) \Rightarrow c(x) = c(y) \tag{5.1}$$

Par extension, nous noterons $c(\Pi_i)$ la classe des exemples de Π_i

Nous avons vu depuis le début de ce chapitre que l'emploi de sous-classes n'est pas toujours avantageux pour la catégorisation. L'algorithme de clustering doit donc avoir un comportement conservateur : il ne doit proposer des sous-classes que si celles-ci sont réellement utiles. Pour cela, nous avons utilisé un algorithme de clustering hiérarchique descendant (voir algorithme 2). L'état initial correspond à une partition avec un cluster par classe (donc sans sous-classe). Lorsque les classes sont homogènes, l'algorithmes doit conserver cet état initial. À chaque étape, le cluster le plus hétérogène (au sens d'une mesure que nous définissons au paragraphe suivant) est divisé en deux sous-clusters par un algorithme de partitionnement. Nous avons choisi d'utiliser k-Means (avec $k = 2$)

Algorithme 2 Clustering hiérarchique descendant avec critère de sélection

Paramètres : une condition θ
Π la partition initiale avec un cluster par classe.
$S = \emptyset$
tant que $S \neq \Pi$ **faire**
  $p = \text{argmin}_{p \in \Pi - S}(||p||)$ (On prend ici le cluster de plus petite norme parmi les clusters qui n'ont pas encore été testés.)
  Appliquer 2-Means sur p. Soient p_1 and p_2 les deux sous-clusters.
  si θ est vérifiée **alors** {La division est acceptée}
    $\Pi = \Pi - p + p_1 + p_2$
    $S = \emptyset$ (Tous les clusters peuvent à nouveau être testés.)
  sinon
    $S = S + p$ (La division du cluster p a été refusée, on l'inclut dans S pour qu'il ne soit pas testé à nouveau.)
  fin du si
fin du tant que

en raison de sa simplicité de mise en œuvre et de sa rapidité d'exécution, mais il peut être remplacé par n'importe quel autre algorithme. Un critère qui dépend des étiquettes de classe de chaque exemple et qui mesure de manière indirecte l'utilité des deux sous-clusters construits est ensuite appliqué pour savoir si la division est acceptée ou non. Si la division n'est pas acceptée, le second cluster le plus hétérogène est choisi pour être testé et ainsi de suite jusqu'à ce que plus aucune division ne soit acceptée.

Pour mesurer l'hétérogénéité des clusters afin de sélectionner le cluster à tester en premier, nous utilisons la norme des prototypes. Elle est égale à la racine carrée de la similarité moyenne inter-documents. Le cluster le plus hétérogène est donc celui qui a la plus petite norme.

$$\begin{aligned} ||c||^2 &= \sum_w c_w^2 = \frac{1}{N_c^2} \sum_w \left(\sum_{d \in c} d_w \right)^2 \\ &= \frac{1}{N_c^2} \sum_w \left(\sum_{d1,d2 \in c} d1_w d2_w \right) \\ &= \frac{1}{N_c^2} \sum_{d1,d2 \in c} sim(d1, d2) \end{aligned} \tag{5.2}$$

L'algorithme que nous venons de présenter correspond à un algorithme de partitionnement hiérarchique descendant classique. La seule nouveauté provient de l'ajout du critère d'acceptation ou de rejet d'une division. C'est lui qui permet à l'algorithme de ne construire que les clusters utiles à la catégorisation. Il doit permettre de différencier les clusters qui appartiennent à des classes hétérogènes (qu'il doit accepter) de ceux qui appartiennent à des classes homogènes (qu'il doit refuser). Il faut pour cela utiliser les étiquettes de classes des documents. L'algorithme est donc *faiblement supervisé*. On peut aussi parler de *clustering discriminant* puisque le but est de construire des clusters qui séparent au mieux les classes.

Nous avons jusqu'à présent simplement déporté la difficulté. Au lieu d'implémenter

un algorithme de clustering spécifique qui différencie automatiquement les classes hétérogènes et homogènes, nous utilisons un algorithme classique. L'évaluation de l'utilité de la création des deux sous-clusters est effectuée a posteriori, après chaque essai de division. Le critère doit évaluer à partir de la liste des clusters si la division que l'on vient de réaliser était utile ou non. Nous avons étudié deux critères pour cette tâche : le premier est purement géométrique et porte sur le positionnement relatif des différents clusters, le deuxième utilise les résultats de classification des exemples. Nous avons également comparé ces deux critères à un comportement basique (critère BASE) qui consiste à accepter systématiquement toutes les divisions. Cet algorithme correspond exactement au clustering hiérarchique classique. Pour ce dernier, il faut de plus spécifier le nombre de clusters souhaités puisqu'il n'y a pas de critère d'arrêt.

5.4.2 Critère PRC : Positionnement Relatif des Clusters

PRC est un critère géométrique basé sur la position des clusters dans l'espace de représentation des documents. Formellement, il s'exprime par :

$$\forall i,j\ c(\Pi_i) = c(\Pi_j) \Rightarrow \exists k,\ c(\Pi_k) \neq c(\Pi_i) \text{ et } \begin{cases} d(\Pi_i,\Pi_k) \leq d(\Pi_i,\Pi_j) \\ d(\Pi_j,\Pi_k) \leq d(\Pi_i,\Pi_j) \end{cases} \tag{5.3}$$

Ce critère signifie que pour chaque paire de cluster (Π_i, Π_j) de la même classe, il doit exister un troisième cluster Π_k d'une autre classe *entre* les deux, c'est-à-dire qui se trouve dans la zone définie par l'intersection des deux sphères de centre Π_i et Π_j et de rayon $d(\Pi_i, \Pi_j)$.

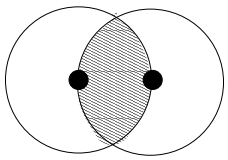


FIG. 5.5 – Illustration de la notion de position *entre* deux clusters : la surface située à l'intersection des deux sphères de centre chaque cluster et de rayon leur distance.

Pour appliquer ce critère, il nous faut connaître à chaque étape les distances entre chaque paire de clusters. Pour limiter les calculs, nous ne testons le critère pour que la nouvelle paire de clusters à valider au lieu de le faire pour tous les couples possibles de clusters. La contrainte globale n'est donc pas garantie puisque l'apparition des deux nouveaux clusters peut invalider le critère pour d'autres couples. Néanmoins, les expériences montrent qu'en pratique elle est pratiquement toujours vérifiée (en effectuant une vérification à la fin de l'algorithme, nous avons trouvé que c'est le cas pour plus de 98% des couples de clusters de même classe).

Ce seul critère n'est toutefois pas suffisant, car il ne gère pas le problème des exemples bruités. Les k-Means ont tendance à diviser un ensemble en un petit cluster très éloigné du centre et un plus grand qui regroupe tous les autres documents. Le petit cluster éloigné du centre a de grandes chances de contenir des exemples bruités. Les deux clusters seront alors très hétérogènes (puisque l'un contient des exemples bruités et l'autre pas), et le critère va donc accepter la division. Ces clusters vont à leur tour permettre de valider de nouvelles divisions qui n'auraient pas été acceptées sans eux (car étant bruités, ils auront tendance à être positionnés dans des zones aléatoires pouvant être entre deux clusters). À la fin du clustering le corpus est constitué d'un très grand nombre de tout petits clusters, ce qui ne correspond pas du tout au comportement souhaité. Pour filtrer ces groupes, nous utilisons un deuxième critère sur la taille : les clusters trop petits (i.e. contenant moins de N_0 documents) ne sont pas autorisés. Ce critère ad-hoc permet effectivement de limiter le nombre de divisions en filtrant les clusters potentiellement bruités.

5.4.3 Critère CD : catégorisation des documents

CD est un critère qui utilise la catégorisation des documents du corpus d'apprentissage pour mesurer l'utilité de la division des clusters.

$$\forall \Pi_i, \quad \left| \left\{ x \in \Pi_i | \exists \Pi_1, \Pi_2 \text{ tel que } \begin{array}{l} c(\Pi_1) = c(\Pi_2) \\ \Pi_1, \Pi_2 \text{ plus proches clusters de } x \end{array} \right\} \right| < |\Pi_i| \quad (5.4)$$

Cette formule exprime le fait que nous voulons minimiser le nombre d'exemples pour lesquels les deux prototypes les plus proches sont de même classe. Si c'est vrai pour un grand nombre de documents, alors il n'est pas nécessaire de distinguer ces deux clusters (voir une illustration dans la figure 5.6). Comme pour le critère PRC, nous ne testons à chaque étape que les deux nouveaux clusters qui viennent d'être créés. Le critère peut donc ne pas être vérifié globalement pour l'ensemble des clusters. Par ailleurs, le critère mis en place est plus strict que la formule (5.4) et plus directement lié à l'algorithme de clustering hiérarchique descendant : les nouveaux clusters sont acceptés si le nombre de documents qui ne vérifient pas le critère est inférieur à $min(|p_1|, |p_2|)$. Ceci permet de supprimer les divisions trop inégales qui consistent en un tout petit et un très grand cluster. Cet ajout permet de limiter implicitement l'influence des exemples bruités lors du clustering. La contrainte supplémentaire sur la taille des clusters est ici inutile.

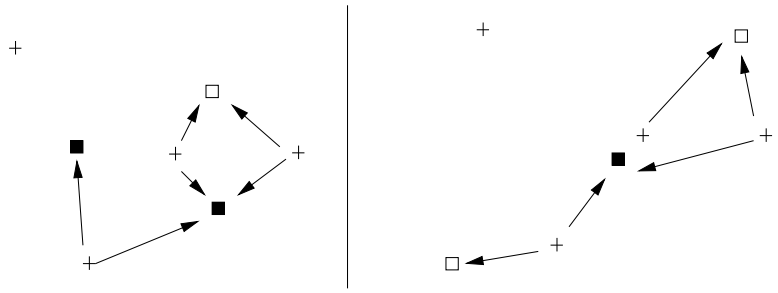


FIG. 5.6 – Critère CD.
Dans la division de gauche, l'exemple du bas ne respecte pas le critère puisque les deux clusters les plus proches de lui sont noirs. Dans la division de droite en revanche, tous les exemples respectent le critère.

5.5 Expériences

5.5.1 Résultats globaux

Nous avons réalisé des expériences sur les corpus 20-Newsgroups (normal, homogène et hétérogène), Mail Center et Spam. Nous avons utilisé les algorithmes Rocchio, k-PPV et SVM. Pour RMP, nous avons implémenté le clustering k-Means, le clustering hiérarchique descendant avec les trois critères BASE, PRC et CD. Pour tous les algorithmes de clustering, nous ne construisons que des clusters "purs" (avec des exemples de même classe). Pour les k-Means et BASE, le nombre de clusters doit être fourni préalablement. Nous utilisons alors le nombre moyen trouvé par les critères PRC et CD. Pour le critère PRC, nous avons optimisé le paramètre N_0 par validation sur une partie du corpus d'apprentissage. La valeur $N_0 = 20$ a été utilisée dans le reste des tests. Pour les algorithmes qui utilisent des tirages aléatoires (donc tous les algorithmes avec clustering, puisqu'ils utilisent k-Means), les tests sont exécutés cinq fois et les résultats donnés sont la moyenne des cinq essais.

Nous avons également appliqué le clustering avec SVM. Dans ce cas, les performances sont systématiquement inférieures à SVM sans clustering. Le résultat est similaire à celui de [OLC+01] et confirme une nouvelle fois l'hypothèse présentée dans la section 5.3 : le clustering n'est pas utile pour les algorithmes discriminants puisque même avec des classes hétérogènes, les surfaces linéaires suffisent à les séparer correctement.

Nous avons également réalisé des expériences avec le corpus Princip et TDT (voir sections 3.3 et 3.5.1). Sur ces deux corpus, l'utilisation de sous-classes semble inutile. Lorsque l'on force le clustering en utilisant le critère BASE, les performances sont systématiquement inférieures au système de base sans sous-classe. Avec les critères PRC ou CD, aucune division n'est acceptée et les classes de départ sont conservées. Ce comportement est bien celui attendu puisqu'ils obtiennent ainsi le clustering le plus efficace :

celui qui consiste à ne pas construire de sous-classes. Cela montre également que, bien que les classes de ces deux applications ne soient pas thématiques, il n'est pas nécessaire d'utiliser les sous-classes. La non-thématicité n'est pas une condition suffisante pour garantir l'utilité du clustering en sous-classes. Nous ne présentons pas plus longuement les résultats sur ces corpus.

Les résultats sont présentés dans le tableau 5.4.

	Newsgroups			Spam	Mail
	normal	hétérogène	homogène		Center
Nombre de classes	20	4	4	2	40
Rocchio	0.81/0.92	0.75/0.92	0.89/0.97	0.77/.	0.51/0.72
k-NN	0.84/**0.94**	0.86/**0.97**	0.93/0.98	0.93/.	0.54/0.65
SVM	**0.87**/0.93	**0.89**/0.97	**0.96**/**0.99**	**0.97**/.	**0.58**/0.71
k-Means	0.81/0.92	0.82/0.94	0.90/0.97	0.96/.	0.53/0.73
Critère BASE	0.81/0.92	0.80/0.94	0.91/0.98	0.95/.	0.56/0.73
Critère PRC	0.81/0.92	0.82/0.94	0.91/0.97	0.96/.	0.54/0.72
Critère CD	0.82/0.92	0.83/0.94	0.91/0.97	0.96/.	0.56/**0.73**

TAB. 5.4 – Mesure de performance : perf-1/perf-2
Les valeurs en italique indiquent le meilleur algorithme pour chaque corpus. Il n'y a pas de mesure perf-2 pour le corpus Spam puisqu'il ne contient que deux classes.

SVM surpasse tous les autres algorithmes et pour tous les corpus avec la mesure perf-1. Ce résultat est conforme à l'ensemble des travaux actuels qui considèrent SVM et le Boosting comme les classifieurs les plus performants. En revanche, et comme nous l'avons déjà vu dans la section 4.4, SVM est moins compétitif sur perf-x en particulier avec le corpus Mail Center.

Les quatre algorithmes de clustering ont un comportement globalement similaire sauf pour le corpus Mail Center. Ce résultat est décevant puisque les critères PRC et CD ont été développés de façon à optimiser le classifieur alors que k-Means et BASE ne sont pas du tout optimisés pour cela. Cela montre que PRC et CD agissent plus comme des critères d'arrêt pour sélectionner le nombre correct de clusters que comme des critères de sélection des bons clusters (puisque les autres algorithmres construisent des clusters d'aussi bonne qualité). Une fois le nombre optimal de sous-classes connu, tous les algorithmes de clustering ont tendance à séparer préférentiellement les classes les plus hétérogènes et donc n'importe lequel convient.

Quel que soit le clustering, RMP est toujours supérieur à Rocchio. Pour les corpus Newsgroups normal et homogène, le clustering construit très peu de clusters (un ou deux) et les performances de RMP sont donc quasiment identiques à celles de Rocchio. Les critères d'arrêt PRC et CD sont donc efficaces pour déterminer le nombre optimal de clusters pour éviter les clusters bruités et le sur-apprentissage. Pour le corpus

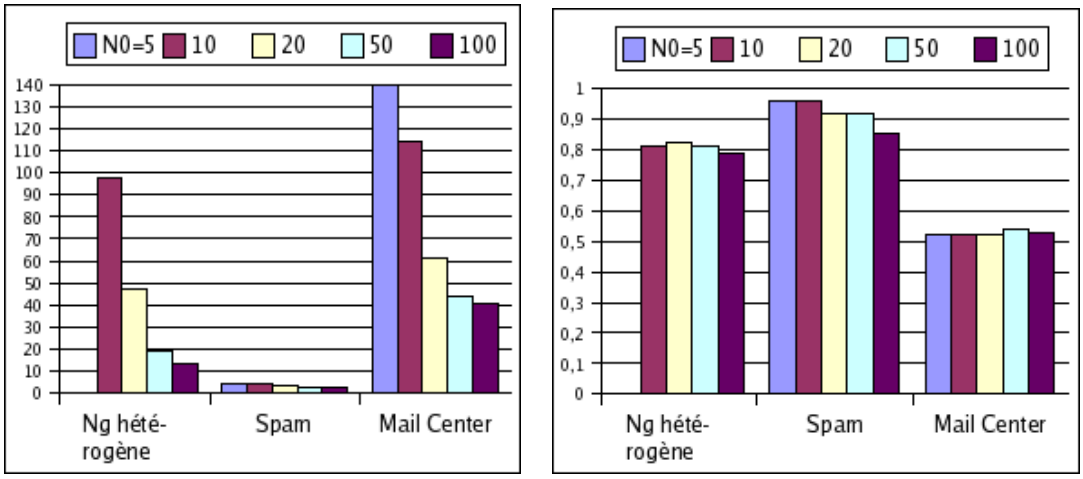
hétérogène, RMP améliore sensiblement Rocchio mais reste très en deça des taux optimaux des expériences de la section 5.3 et des taux de SVM ou k-PPV. Même avec un corpus construit explicitement pour mesurer l'apport des sous-classes, RMP n'est pas compétitif face à SVM. Il n'est utile que dans les applications où Rocchio a déjà de bonnes performances. Pour les corpus Spam et Mail Center, l'amélioration de RMP est beaucoup plus sensible. Il surpasse k-PPV pour les deux corpus et même SVM sur perf-2 avec Mail Center. Nous analyserons plus en détail dans la suite les raisons de ce succès.

L'amélioration de RMP par rapport à Rocchio sur perf-2 est toujours inférieure à 1% alors qu'elle peut atteindre 19% sur perf-1. Ceci s'explique par le fait que les critères PRC et CD ont un fonctionnement de type discriminant : ils cherchent à construire des clusters qui modifient les frontières entre les classes. Nous avons vu dans la section 4.4 que les algorithmes discriminants ne sont pas performants avec les mesures perf-x, il est donc logique que RMP n'améliore pas sensiblement Rocchio sur cette mesure. Il permet, en revanche, d'améliorer les faibles performances de Rocchio sur perf-1 tout en conservant la compétitivité de ce dernier sur perf-x.

Le critère PRC nécessite de spécifier la valeur du paramètre N_0 avant de commencer le clustering. Pour mesurer l'influence de ce paramètre, nous avons testé le clustering avec plusieurs valeurs possibles du paramètre (entre 5 et 100) et mesuré les performances pour chacune d'entre elles. Les résultats sont indiqués dans la figure 5.7. On peut voir tout de suite que le nombre de clusters est très dépendant de ce paramètre et les taux de performance également. De plus, la valeur optimale du paramètre est différente pour chaque corpus : 20 pour Newsgroup, 5 pour Spam, 50 pour Mail Center. L'utilisation du critère PRC nécessite donc une phase de validation préalable pour déterminer la meilleure valeur de N_0. PRC semble moins performant que CD (en particulier sur le corpus Mail Center) et son efficacité dépend d'un paramètre qu'il faut optimiser pour chaque application. Pour ces deux raisons, nous avons finalement abandonné le critère PRC et nous ne conservons pour la suite de nos tests que le critère CD.

Nous avons également réalisé des tests sur le corpus Reuters 2000[1]. Dans ce corpus, les documents sont attribués à plusieurs catégories. Les mesures perf-x n'étant pas adaptées à un cadre de filtrage, nous avons utilisé la mesure F_1 (voir section 2.6.1). Les résultats sont relativement inattendus (voir tableau 5.5). SVM est le moins performant des algorithmes et k-PPV le meilleur ! RMP permet d'améliorer Rocchio de presque 20%. Le clustering fournit ainsi 400 clusters (pour 100 classes). Ces résultats relativement étonnants s'expliquent si l'on suppose que les classes ne sont pas linéairement séparables. k-PPV est alors particulièrement adapté, alors que nous avons utilisé dans nos tests la version linéaire de SVM. Cela explique également l'amélioration de RMP par rapport

[1] `http://about.reuters.com/researchandstandards/corpus/`



Nombre de clusters perf-1

FIG. 5.7 – Comportement de l'algorithme PRC en fonction de N_0.

à Rocchio. Il semble donc que RMP peut également être utile dans le cas de classes thématiques mais difficiles à cerner. Des expériences supplémentaires avec ce corpus encore peu exploité sont nécessaires pour analyser plus en détails les raisons de ces résultats.

	Rocchio	30-PPV	SVM	RMP CD
F_1	0.50	0.61	0.45	0.59

TAB. 5.5 – Résultats sur le corpus Reuters 2000

5.5.2 Analyse des clusters obtenus

Nombre et qualité des clusters

Les performances de RMP sont directement dépendantes de la qualité du clustering. Il est donc utile d'analyser les clusters obtenus pour s'assurer que le comportement du clustering est conforme à notre attente, c'est-à-dire que les sous-thèmes des classes hétérogènes sont bien détectés mais pas ceux des classes homogènes.

Nous pouvons vérifier ceci sur le corpus Newsgroups avec les deux bases homogènes et hétérogènes. Dans ces expériences, le nombre optimal de clusters est connu à l'avance. Sur la base homogène, nous nous attendons à ce que le clustering soit inutile et que aucune sous-classe ne soit générée. Sur la base hétérogène, le clustering devrait retrouver les 20 groupes de départ. Avec le critère DC, RMP obtient 5.4 clusters sur la base homogène, c'est-à-dire 1.4 de plus que les quatre classes de départ. Sur la base hétérogène, il obtient 19 clusters, ce qui est très proche des 20 optimaux. Le critère a donc bien

détecté une différence entre les deux corpus alors que les documents et les vingt groupes de base sont les mêmes.

Nous pouvons, de plus, vérifier que les clusters correspondent effectivement à des thématiques différentes avec le corpus Newsgroups hétérogène, puisque les clusters optimaux sont connus (ce n'est pas possible de le faire avec le corpus homogène, puisque le clustering ne construit aucun cluster ou presque). Chaque classe devrait contenir cinq clusters de 600 documents qui correspondent à un groupe de discussion. Dans le clustering obtenu, la moitié des clusters sont effectivement formés d'environ 600 documents (entre 500 et 700) constitués à plus de 90% de documents d'un même groupe de discussion. Dans un autre quart, ils sont formés de 200 documents issus d'un même groupe de discussion (ces clusters sont homogènes mais tous les documents de la thématique ne sont pas regroupés). Dans le dernier quart, ils sont formés de plus de 1000 documents qui regroupent deux groupes de discussion souvent d'une thématique proche (comme il y a 4 thèmes principaux et 5 sous-groupes par classe, chaque classe contient en général deux groupes du même thème principal). Le clustering n'est donc pas conforme aux catégories initiales, mais la séparation thématique est néanmoins correctement effectuée.

Pour le corpus Spam, l'analyse des clusters se révèle aisée. Les quatre clusters formés correspondent pour les deux classes à la séparation entre les courriers en anglais et en français. Il y a donc 43 spams en français pour 457 en anglais et 488 courriers légitimes en français pour 472 en anglais. Le clustering a donc permis ici de séparer les langues sans algorithme spécifique.

Pour le corpus Mail Center, l'analyse des 70 clusters obtenus est plus difficile. Il est possible d'obtenir le profil de chaque cluster, c'est-à-dire la liste des termes avec leur poids associé, mais cela ne permet pas toujours de déterminer leur pertinence. Nous avons pu trouver une séparation thématique correcte pour environ la moitié des clusters. Par exemple, une classe correspond aux questions relatives à la procédure permettant d'insérer une image dans une page. Les documents de la classe ont été séparés en trois, toutes les questions correspondant à la même réponse-type. Le premier cluster correspond à des questions sur les formats d'image autorisés, le deuxième à des questions sur le moyen de modifier une image déjà insérée, le troisième sur la taille maximale autorisée d'une image. Quelques clusters correspondent également à des différences de format : les mails en HTML d'une part et les mails en texte standard d'autre part. Bien que cette séparation ne soit pas thématique, elle est utile au classifieur pour limiter l'effet dû à ces formats. Malheureusement, pour à peu près la moitié des clusters, il n'a pas été possible d'identifier de thèmes pour expliquer leur formation.

Une deuxième manière de s'assurer que le nombre de clusters est correct est de faire une recherche exhaustive. Pour cela, nous avons utilisé le clustering avec le critère BASE. Nous avons fait varier le nombre de clusters entre 1 et 200 en mesurant le taux

de performance dans chaque cas. La figure 5.8 donne une représentation graphique des résultats. Le critère CD semble trouver un nombre de clusters légèrement inférieur à la valeur optimale, mais avec un taux de performance égal au meilleur taux de l'algorithme BASE. Globalement, nous considérons que le clustering parvient à trouver un nombre approprié de clusters.

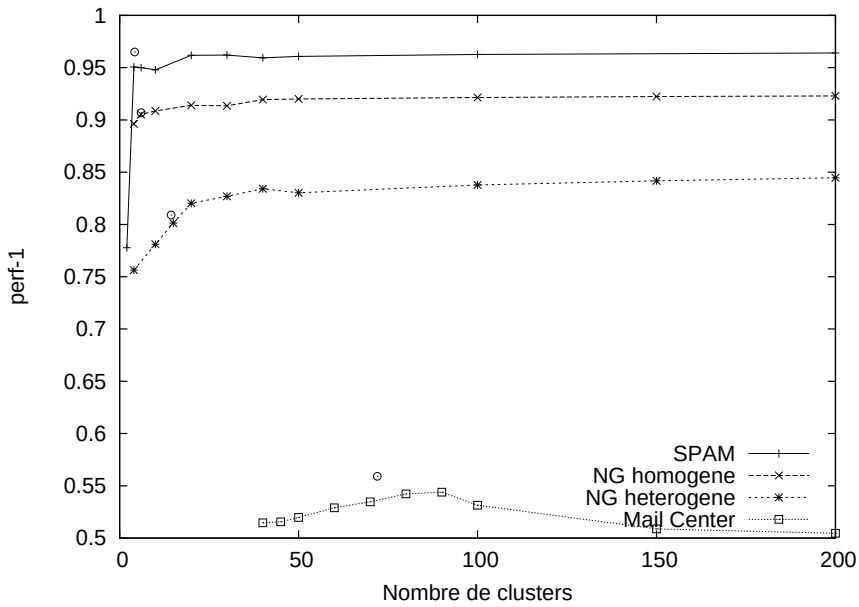


FIG. 5.8 – Performance de RMP avec critère BASE en fonction du nombre de clusters. Les ronds représentent le nombre de clusters et le taux de performance pour RMP avec critère CD.

Variabilité du clustering

Dans le clustering k-Means, la partition finale dépend des centres initiaux, qui sont choisis aléatoirement. Le résultat est donc différent à chaque exécution. Comme nous utilisons les k-Means pour diviser un cluster en deux, le clustering RMP est lui aussi soumis à des variations aléatoires. Nous souhaitons étudier la variabilité du nombre de clusters et du taux de performance dans RMP.

	Spam	Newsgroups hétérogène	Mail Center
Nombre de clusters	4 ± 0	19.2 ± 2.1	70.6 ± 2.1
Performance	0.96 ± 0	0.83 ± 0.002	0.56 ± 0.02

TAB. 5.6 – Moyenne et variance du nombre de clusters et du taux de perfomance avec RMP sur cinq essais.

La moyenne et la variance du nombre de clusters et du taux de performance sur cinq essais pour chaque corpus Spam, Newsgroups hétérogène et Mail Center sont indiquées dans le tableau 5.6. Avec le corpus Spam, le clustering est très facile. Les deux langues sont détectées correctement dans les cinq essais. L'aspect aléatoire des k-Means n'apparaît pas lorsque les clusters sont très facilement différenciables comme ici. Avec le corpus Newsgroups hétérogène, le nombre de clusters et les performances varient très peu. Avec le corpus Mail Center, la variabilité est plus importante. Le nombre de clusters reste relativement stable mais les performances varient de plus de 4%.

Nous retrouvons ici le fait que l'algorithme de clustering avec critère CD n'est pas meilleur que n'importe quel autre. La variabilité des k-Means se retrouve donc dans les variations de taux de performance. En revanche, le critère permet effectivement de trouver le nombre approximativement correct de clusters avec une faible variance. À nouveau, le critère CD semble plus utile en tant que critère d'arrêt pour déterminer le nombre de clusters que pour optimiser le partitionnement. Il permet donc de déterminer la meilleure complexité à utiliser pour modéliser les classes. Une fois, la complexité fixé, tous les classifieurs peuvent apprendre les paramètres de ce modèle.

La qualité des clusters semble donc dépendre beaucoup plus de l'algorithme qui divise les clusters en deux que des critères qui décident de l'acceptation d'une division. Il semble donc possible de remplacer les k-Means par un autre algorithme plus performant et moins fluctuant tout en conservant le critère CD qui permet de choisir le nombre de clusters.

5.5.3 Influence de la taille du corpus

Rocchio a été initialement développé pour interagir avec l'utilisateur lors d'une recherche d'informations (feedback : voir section 2.3.1). Il est donc particulièrement adapté lorsque le corpus d'apprentissage contient peu d'exemples. Il surpasse en général les algorithmes plus complexes comme SVM dans ce cas là. Les algorithmes de clustering que nous utilisons dans RMP nécessitent un plus grand nombre d'exemples pour pouvoir créer des clusters statistiquement cohérents. Il n'est donc pas assuré que RMP se comporte aussi bien que Rocchio avec de petits corpus.

Pour vérifier ceci, nous avons réalisé une expérience supplémentaire avec le corpus Newsgoups (c'est le corpus qui contient le plus grand nombre de documents). Nous avons comparé les performances de Rocchio, RMP, SVM et k-PPV pour plusieurs tailles de corpus d'apprentissage. Les résultats présentés dans la figure 5.9 sont conformes à nos suppositions. Avec de très petits corpus, k-PPV et SVM ne peuvent pas prédire la classe avec suffisamment de précision alors que RMP et surtout Rocchio sont encore utilisables. En revanche, avec des corpus plus grands, Rocchio semble incapable d'utiliser toute l'information présente dans les textes et sa performance ne change quasiment pas

entre 1000 et 10000 documents. RMP continue à s'améliorer mais moins que k-PPV et SVM qui le dépasse rapidement.

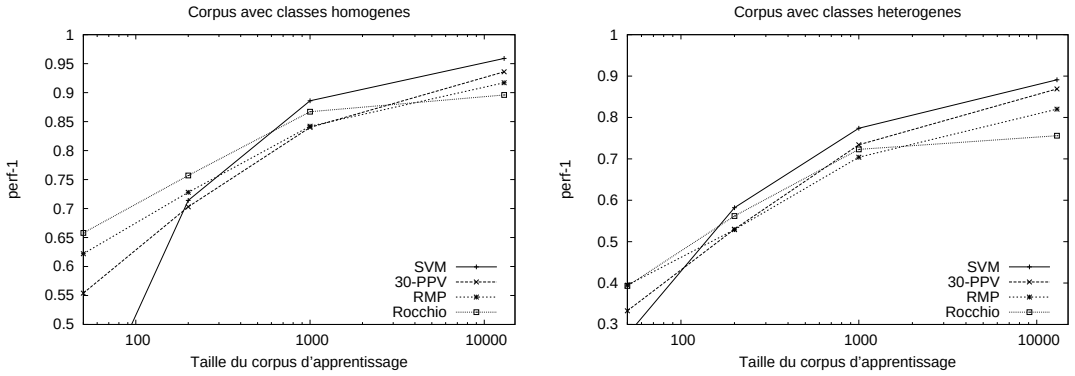


FIG. 5.9 – Taux de performance en fonction de la taille du corpus d'apprentissage.

Jusqu'à présent, nous avions montré que Rocchio pouvait être avantageusement remplacé par RMP dans tous les cas. Si les classes ne sont pas hétérogènes, cela n'a pas d'influence et si elles le sont, le taux de performance sera amélioré. Nous venons de voir que RMP peut dégrader les performances de Rocchio si le corpus est trop petit. Il faut donc utiliser RMP uniquement dans le cas où le corpus contient suffisamment de documents par rapport au nombre de classes. Malheureusement, la taille critique est très dépendante du corpus.

5.6 Conclusion

Les nouvelles applications de la CT ont des caractéristiques particulières : les données sont très bruitées, les classifieurs sont utilisés en mode semi-automatique ou encore les catégories sont non thématiques. Les deux premières spécificités font que SVM ou le Boosting ne sont pas toujours les algorithmes les plus adaptés pour ces applications. Rocchio est plus robuste au bruit et plus performant en mode semi-automatique. En revanche, les classes non thématiques risquent d'être une gêne importante pour lui.

Au cours de ce chapitre, nous avons étudié en détail le problème des classes multi-thématiques. Pour le résoudre, nous proposons d'utiliser un algorithme de clustering qui détermine de manière faiblement supervisée les sous-thèmes de chaque catégorie. Les sous-classes servent ensuite à modéliser les classes : plus elles sont nombreuses, plus le modèle des classes est complexe. Le clustering est donc un moyen de déterminer la meilleure complexité. Les classifieurs apprennent ensuite les poids optimaux pour un modèle fixé. Nous avons montré que la détection de sous-classes n'est utile que dans le cas où les classes de départ sont thématiquement très hétérogènes. Les algorithmes

discriminants ne semblent pas gênés par le manque d'homogénéité dans les classes. Le clustering n'est donc utile que pour les algorithmes génératifs qui ne sont pas capables de modéliser directement des classes complexes.

Nous avons développé un algorithme de clustering qui permet de détecter des clusters lorsque les classes sont hétérogènes et qui conserve les classes d'origine dans le cas contraire. Cet algorithme est utilisé en combinaison avec Rocchio qui est l'un des classifieurs génératifs le plus performant à l'heure actuelle.

Les expériences réalisées montrent que le clustering parvient effectivement à différencier les classes hétérogènes des classes homogènes. Dans ce cas-là, RMP (Rocchio Multi-Prototypes) est effectivement plus performant que Rocchio. Dans le cas contraire, les différences sont quasiment inexistantes. RMP permet surtout d'améliorer la mesure perf-1 tout en conservant les scores élevés de Rocchio sur perf-x. Il parvient ainsi à se rapprocher sensiblement des résultats de SVM sur perf-1 tout en restant très largement supérieur à SVM avec perf-x. Nous avons montré que l'algorithme CD parvient à sélectionner correctement le nombre optimal de clusters. Par contre, une fois le nombre correct de clusters obtenu, il ne semble pas plus performant que n'importe quel autre algorithme de clustering. Cela signifie qu'il doit être possible d'améliorer RMP en utilisant un algorithme de clustering plus évolué que k-Means tout en conservant ce critère pour calculer le nombre correct de clusters. Enfin, pour que l'utilisation de sous-classes soit réellement intéressante, il est nécessaire d'utiliser avec un corpus de taille importante.

Chapitre

6

Détection des changements de concepts

Sommaire du chapitre

6.1 Introduction

Le modèle que nous avons étudié jusqu'à présent ne considère que les corpus statiques. Dans une première phase de mise en route, un corpus d'apprentissage est constitué et le classifieur apprend à discriminer les catégories à partir de ces exemples. Une fois l'apprentissage effectué, il n'est plus remis en cause et le classifieur est utilisé pour catégoriser de nouveaux documents, et ceci aussi longtemps que l'application est exploitée.

Dans de nombreux cas, il est préférable de réactualiser régulièrement le classifieur en lui fournissant de nouveaux exemples, plus récents, qui sont intégrés au fur et à mesure de leur arrivée dans le corpus d'apprentissage. Le classifieur est alors modifié en fonction de ces documents. Cette mise à jour permet de tenir compte des mots

nouveaux qui n'apparaissaient pas auparavant (comme les noms de personnes ou les termes scientifiques) et qui sont souvent très discriminants pour la classification. Elle permet également de prendre en compte les changements dans la structure même de la tâche lorsque des catégories sont ajoutées, supprimées ou simplement modifiées.

C'est le cas par exemple de l'application classique du classement des dépêches de presse. Le vocabulaire change très rapidement et il faut mettre à jour les statistiques lexicales. Dans le problème du filtrage de Spam, certains sous-thèmes sont très stables et d'autres apparaissent et disparaissent rapidement (par exemple, les courriers proposant la vente de Viagra étaient très fréquents il y a un ou deux ans, ils sont aujourd'hui marginaux). Le Mail Center est un exemple d'application où les catégories changent au fur et à mesure du temps : de nouvelles problématiques apparaissent, qui génèrent de nouvelles réponses-types et donc de nouvelles classes (par exemple lorsque l'entreprise lance des opérations promotionnelles limitées dans le temps ou lorsque des problèmes techniques temporaires surgissent).

Le modèle d'apprentissage que nous allons considérer est le suivant : un nouveau classifieur est appris à chaque instant t à partir de tous les exemples passés. Entre t et $t+1$, un nouveau lot de documents à classer arrive. Ils sont ensuite incorporés au corpus d'apprentissage, une fois leur classe connue.

La gestion temporelle d'un corpus d'apprentissage pose deux problèmes distincts. La première difficulté, que nous ne traiterons pas ici, concerne l'utilisation d'algorithmes d'apprentissage incrémentaux. Les algorithmes incrémentaux utilisent le classifieur appris à l'étape t et uniquement les exemples arrivés entre t et $t+1$ pour apprendre le nouveau classifieur de l'étape $t+1$. Un algorithme incrémental idéal doit fournir à chaque étape un classifieur identique à celui qui aurait été appris si tous les exemples avaient été fournis dès le début (voir par exemple l'algorithme incrémental pour les arbres de décisions de Utgoff [Utg89]). Le but des algorithmes incrémentaux est de pouvoir réactualiser un classifieur d'une étape à l'autre plus rapidement que par un réapprentissage complet. Il s'agit bien d'un problème d'efficacité computationnelle et non pas d'améliorer la justesse des prédictions. Le problème est trivial pour k-PPV puisque aucun traitement n'est réalisé pendant la phase d'apprentissage. Rocchio est facilement adaptable au cas incrémental (puisque la moyenne d'une série de nombres est calculable incrémentalement). SVM a également été adapté pour le cas incrémental par L. Ralaivola et F. d'Alché-Buc [RdB01]. Nous considérons ce problème comme résolu, au moins pour les algorithmes que nous allons utiliser.

La deuxième difficulté concerne la qualité du classifieur. Le corpus d'apprentissage grossit indéfiniment au fur et à mesure de l'arrivée des nouveaux documents. Comme sa taille augmente à chaque ajout, chaque exemple fournit une contribution de plus en plus marginale dans la constitution du classifieur. Au bout d'un certain temps, le corpus

d'apprentissage est si grand que l'ajout de nouveaux documents ne modifie quasiment plus le classifieur. Dans le cas où les concepts ne changent pas au cours du temps, cela ne pose pas de problème et l'accumulation d'exemples permet d'apprendre les profils avec de plus en plus de fiabilité. En revanche, si la tâche de classification change au cours du temps, comme c'est le cas dans les exemples d'applications que nous avons présentés, le classifieur est de moins en moins performant, car les nouveautés ne sont pas prises en compte sur leur importance réelle, puisque les anciens exemples ont autant de poids que les nouveaux.

Pour résoudre ce problème, la solution souvent préconisée consiste à utiliser une *fenêtre glissante* sur la liste des documents pour obtenir le corpus d'apprentissage (voir la figure 6.1). La fenêtre peut être définie temporellement (ie. les documents de moins de x jours sont pris en compte) ou par la taille du corpus (ie. les x plus récents documents sont pris en compte). Plus la fenêtre est large, plus il y a de documents dans le corpus d'apprentissage et donc plus les statistiques sont fiables. À l'inverse, plus la fenêtre est petite, plus le classifieur est réactif aux changements de vocabulaire ou de concepts. Le choix de la taille de la fenêtre est donc un compromis entre ces deux comportements.

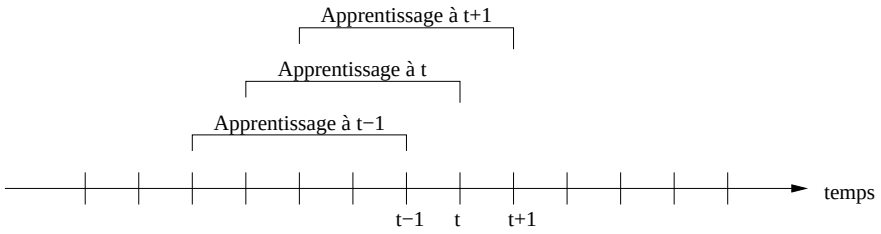


FIG. 6.1 – Illustration de la notion d'apprentissage sur une fenêtre glissante. L'apprentissage est réalisé uniquement avec les exemples des x derniers lots (ici $x = 4$) et non pas l'ensemble du corpus.

Il est possible de conserver les avantages des deux solutions en utilisant une fenêtre à taille variable. Elle est agrandie au maximum lorsque le corpus est stable et fortement réduite lorsqu'un changement est détecté. Il faut pour cela mettre en place un algorithme qui détecte les changements de concepts afin de modifier la taille de la fenêtre en conséquence.

C'est cette problématique que nous étudions dans ce chapitre. Nous présentons en première partie un état de l'art des algorithmes actuels pour traiter ce problème. Dans la section 6.3, nous décrivons une nouvelle méthode fondée sur l'algorithme de clustering présenté dans le chapitre précédent. Les expériences décrites dans la section 6.4 mettent en évidence les avantages et les inconvénients de cette nouvelle méthode.

6.2 État de l'art

Divers travaux théoriques [HL94, BL97, FM97] fournissent les vitesses maximales de variation qu'un concept peut subir pour qu'il reste apprenable avec un taux d'erreur fixé. Cette vitesse est généralement définie dans un cadre probabiliste par la distance entre les distributions de probabilité qui génèrent les données à l'instant t et $t+1$. Ces limites théoriques ne donnent pas d'indication sur la façon d'implémenter un algorithme capable de résoudre ce problème. Il faut alors se tourner vers les travaux expérimentaux, qui proposent des heuristiques et les comparent expérimentalement à d'autres algorithmes. Le principe général est d'utiliser une fenêtre glissante de taille variable. Plusieurs indicateurs qui dépendent généralement indirectement de la stabilité du corpus d'apprentissage sont contrôlés à intervalle régulier. S'ils varient de façon anormale, un changement de concept est suspecté et la taille de la fenêtre est réduite. D'après R. Klinkenberg [KR98], ces indicateurs se classent dans trois grandes catégories :

- les mesures de performance (par exemple le taux de bonne classification) : pour utiliser cet indicateur, il faut avoir accès aux réponses correctes pour chaque document.
- les propriétés du classifieur (par exemple en mesurant la complexité des règles obtenues) : cet indicateur ne peut être utilisé que si le classifieur a une représentation interne des classes qui est plus ou moins interprétable.
- la description des données (par exemple la distribution des classes ou des mots).

Le premier type d'indicateur est le plus employé. R. Klinkenberg et I. Renz [KR98] l'utilisent dans son principe le plus simple. Le rappel et la précision sont mesurés à chaque lot, le taux moyen de bonne classification et sa variance sont également calculés à partir des résultats sur les N lots précédents. Si l'une des deux mesures est très inférieure à la moyenne, un glissement de concept est détecté et la fenêtre est réduite de 20%. Si les deux mesures sont très inférieures, un changement de concept est détecté et la fenêtre est réduite à son minimum (c'est-à-dire le dernier lot d'exemples, tous les précédents étant oubliés). Si aucun changement n'est détecté, la fenêtre augmente en incorporant le nouveau lot.

Dans le système *FLORA* [WK96] (apprentissage de règles de décision), G. Widmer et M. Kubat utilisent les deux premiers types d'indicateurs. Pour chaque règle, le taux de bonne classification et la couverture de la règle sont calculés. Suivant leur valeur et leur variation, la taille de la fenêtre est agrandie ou réduite.

C. Lanquillon [Lan99] utilise le deuxième type d'indicateur dans un contexte de filtrage. Un changement de concept est suspecté lorsque le pourcentage d'exemples fortement négatifs (c'est-à-dire le nombre d'exemples dont la similarité avec le profil de la classe est très faible) dépasse un certain seuil. Cette méthode permet de détecter les changements sans même connaître les étiquettes de classe de chaque exemple.

R. Klinkenberg et T. Joachims [KJ00] proposent une toute autre approche. Il s'agit simplement d'optimiser la taille de la fenêtre à chaque instant en mesurant le taux de performance pour toutes les tailles possibles. Il suffit ensuite de choisir la taille qui fournit le meilleur taux. Cette solution est possible grâce à l'utilisation d'une technique spécifique au classifieur SVM qui permet de faire un apprentissage sur le corpus complet et d'estimer avec une très faible erreur le taux de performance que le classifieur aurait eu si des exemples d'apprentissage n'avait pas été présent et cela sans refaire intégralement l'apprentissage (ce qui serait trop long). Dans cette méthode, il n'y a pas, à proprement parler, de détection de changements de concepts.

Une dernière approche pour résoudre ce problème consiste à mesurer l'utilité des exemples pour la classification. Les travaux de M. Salganicoff [Sal93] en offrent une bonne illustration. Le classifieur utilisé est k-PPV, ce qui permet de gérer des indicateurs par document. Le taux de bonne classification et le nombre de voisins de même classe sont sauvegardés pour chaque exemple. Si l'une des deux valeurs tombe en dessous d'un certain seuil, l'exemple est supprimé. Ce principe a également été utilisé lors des travaux sur TDT [NDb] pour mesurer l'activité des catégories. Chin Chen et al. ont ainsi proposé la théorie du vieillissement (aging theory) [CCSC03] qui généralise les travaux de J. Allan [APL98] et de Y. Yang [YPC98] sur l'attribution d'un poids décroissant en fonction du temps aux documents et aux clusters.

À l'exception des travaux de Salganicoff, la fenêtre utilisée est toujours globale. Si un changement de concept est détecté pour une classe c_i, tous les exemples plus vieux que le changement de concept, qu'ils appartiennent ou non à l'ancien concept, sont supprimés. Il serait pourtant préférable de ne supprimer que les exemples qui concernent l'ancien concept et conserver tous les autres Cet inconvénient provient du fait que tous les algorithmes présentés ici ne cherchent pas réellement à découvrir quel est l'ancien et quel est le nouveau concept, mais simplement à détecter que quelque chose a changé.

6.3 Détection par le taux d'activité des clusters

Dans le chapitre précédent, nous avons présenté un algorithme de clustering qui améliore les performances de Rocchio dans le cas où les classes sont constituées de plusieurs sous-thèmes. Le problème est identique dans le cas de la détection de changements de concepts. Lorsqu'il y a un changement de concept, cela signifie que la classe sera constituée de deux sous-classes, l'une correspondant à l'ancien et l'autre au nouveau concept.

Exactement comme pour RMP, si les deux concepts sont proches (classe homogène), la détection du changement est inutile. Elle ne doit intervenir que dans le cas où les deux concepts sont très différenciés, au point de causer des erreurs de classification. L'algorithme de clustering que nous avons développé permet précisément de détecter ce

genre de situations. Nous pensons donc qu'il est intéressant de l'utiliser pour la détection de changements de concepts.

6.3.1 Le taux d'activité

Notre méthode utilise uniquement le troisième type d'indicateur, c'est-à-dire une information sur la distribution des exemples. Nous considérons qu'un changement de concept a eu lieu si un cluster ne contient que des vieux documents. Ce cluster correspond alors à un ancien concept qui n'existe plus. Il est donc inutile et peut être supprimé. Cette méthode est proche de la théorie du vieillissement de Chin Chen, qui a été proposé au moment où nous développions notre propre solution, mais la gestion temporelle est faite pour chaque cluster plutôt que pour chaque classe.

À chaque cluster c_i est attribué un taux d'activité $\gamma(c_i)$ compris entre 0 et 1. Cette pondération est une mesure du degré d'activité du cluster, qui permet d'estimer le nombre de documents qui vont apparaître dans le futur à proximité de ce cluster.

$$\gamma(c_i) = f\left(\frac{1}{M} \sum_{d_j \in \text{Newest}_M(c_i)} date(d_j)\right) \tag{6.1}$$

avec $\text{Newest}_M(c_i)$ l'ensemble des M documents les plus récents de c_i et f : temps $\mapsto [0,1]$ une fonction croissante.

Dans la formule (6.1), l'activité d'un cluster est définie en calculant la date moyenne des M documents les plus récents. On appellera par extension *date du cluster* cette date moyenne prise sur les M documents les plus récents. Si $M = 1$, la date du cluster est égale à la date du dernier document arrivé. Si $M = \infty$, elle est égale à la moyenne de tous les documents. Si M est trop petit, l'estimation n'est pas assez fiable, car basée sur trop peu d'éléments. Si M est trop grand, l'aspect temporel est faussé car des éléments trop vieux sont pris en compte. Le taux d'activité est ensuite calculé comme fonction croissante de cette date. Nous avons essayé trois types de fonctions : binaire, linéaire et exponentielle. Elles sont paramétrées par un coefficient α qui spécifie la date à laquelle $\gamma(c_i)$ vaut 1/2. Le coefficient m correspond à la date à laquelle $\gamma(c_i)$ vaut 1, c'est-à-dire la date du document le plus récent.

$$\begin{aligned} m &= \max_{d_j \in \mathcal{D}}(date(d_j)) \\ f_1(c_i) &= sign(date(c_i) - \alpha)) \\ f_2(c_i) &= \tfrac{date(c_i)+m-2\alpha}{2(m-\alpha)} \\ f_3(c_i) &= \exp\left(-\log(2)\tfrac{m-date(c_i)}{m-\alpha}\right) \end{aligned} \tag{6.2}$$

Ces pondérations $\gamma(c_i)$ sont ensuite utilisées lors de la classification comme une probabilité a priori : le degré de pertinence d'un cluster est le produit de la similarité et du taux d'activité.

$$class(d_j) = \text{argmax}_{c_i}(sim(d_j, c_i) * \gamma(c_i)) \tag{6.3}$$

Nous avons jusqu'ici présenté le taux d'activité pour RMP, permettant de mesurer le degré d'activité de chaque cluster. Lorsqu'il n'y a pas de clustering, il est possible d'utiliser cette méthode directement sur les classes. Elle permet alors de détecter les classes qui ne sont plus utilisées, mais pas les changements de concept qui ont lieu sur une partie seulement d'une classe.

6.3.2 Commentaires

Le principe du taux d'activité avec $M = \infty$ est proche de celui de la fenêtre glissante. Dans cette dernière, plus un exemple est vieux, plus son poids est faible (dans la plupart des techniques avec fenêtre, la pondération est binaire mais ce n'est pas obligatoirement le cas). Le taux d'activité est identique mais appliqué à chaque cluster au lieu d'être appliqué à chaque exemple : plus l'âge moyen des exemples du cluster est élevé, plus son poids est faible. Ici, les exemples ne sont pas gérés indépendamment et tous les exemples d'un même cluster, qu'ils soient anciens ou récents, reçoivent le même poids.

Lorsque M est plus petit, le comportement du taux d'activité s'éloigne alors de celui d'une fenêtre glissante. La pondération d'un cluster est calculée non plus avec tous les documents mais seulement les plus récents. Les exemples plus anciens interviennent toujours pour calculer le profil du cluster mais pas pour le calcul du taux d'activité. Il est donc tout à fait possible de se retrouver dans une situation où des documents moyennement récents sont supprimés car appartenant à un cluster avec peu de documents très récents alors que des documents plus anciens sont conservés car appartenant à un cluster avec des des documents très récents.

À l'exception de Salganicoff (qui gère une fenêtre par exemple), aucun algorithme existant n'est capable de déterminer exactement les exemples concernés par un changement de concept. Lorsqu'un changement est détecté, la seule solution est de supprimer tous les exemples de la classe. L'utilisation du clustering permet de déterminer plus précisément les exemples à supprimer et de conserver les autres exemples, parfois plus anciens mais qui sont toujours utiles pour représenter d'autres concepts. Ceci est un atout important pour les applications où les changements de concepts apparaissent régulièrement mais sur des thèmes limités à chaque fois.

Un autre point important de notre système qui se retrouve dans la théorie du vieillissement et dans les autres travaux sur TDT est que chaque cluster bénéficie d'un coefficient compris entre 0 et 1. Il n'y a pas de décision binaire sur l'inclusion ou l'exclusion d'un document dans le corpus d'apprentissage. L'affaiblissement des exemples est progressif. Cette souplesse permet généralement des taux de performance plus élevés.

Le taux d'activité vise à estimer la quantité de documents qui va apparaître dans les prochains cycles. Dans notre approche, ce taux résulte du calcul de la date moyenne des derniers documents arrivés. Il est possible d'utiliser des méthodes plus complexes pour obtenir une estimation plus fiable tel que les modèles Poissoniens [Cox62] ou des modèles prenant en compte l'aspect cyclique des données. Dans la théorie du vieillissement de Chin Chen par exemple, l'estimation tient compte du fait que la classe correspond à un évènement court ou long.

6.4 Expérimentations

A travers ces expérimentations, nous cherchons à répondre à plusieurs questions : (i) le taux d'activité est-il plus performant qu'une simple fenêtre, (ii) peut-il être appliqué avec succès à des algorithmes sans sous-classes, et (iii) sur quels types de changements de concept est-il le plus performant ?

Pour répondre aux deux premières questions, nous avons utilisé le corpus Mail Center. Les documents de ce corpus sont datés et représentent approximativement un mois de correspondance. Nous pensons que ce corpus contient effectivement des changements de concept même s'il est impossible de les localiser précisément. C'est la première fois, à notre connaissance, qu'un corpus réel est utilisé pour tester des algorithmes de détection de changements (dans le corpus TDT, il ne s'agit pas de détecter les changements de concepts mais de gérer l'obsolescence des classes).

Il n'est malheureusement pas possible d'utiliser ce corpus pour répondre à la troisième question puisque nous n'avons pas d'informations sur les changements de concept. Nous avons donc eu recours à des corpus artificiels pour créer explicitement des changements de concept.

6.4.1 Sur le corpus Mail Center

Pour les expérimentations sur le corpus Mail Center, nous avons utilisé les 2151 premiers documents comme base d'apprentissage et les 215 suivants comme base de test.

Dans un premier temps, nous avons mesuré les performances de RMP pour plusieurs valeurs de M et pour les trois fonctions f proposées à la section précédente, afin de trouver le meilleur paramétrage de l'algorithme. La figure 6.2 montre les résultats obtenus. Sans surprise, la méthode exponentielle est la plus performante, suivie par la méthode linéaire puis la méthode binaire. La valeur optimale de M est un compromis entre une meilleure robustesse au bruit (M grand) et une bonne réactivité (M petit). Pour ce corpus, la valeur optimale semble être $M = 20$. Mais cette valeur dépend du corpus et doit donc être réajustée pour chaque application.

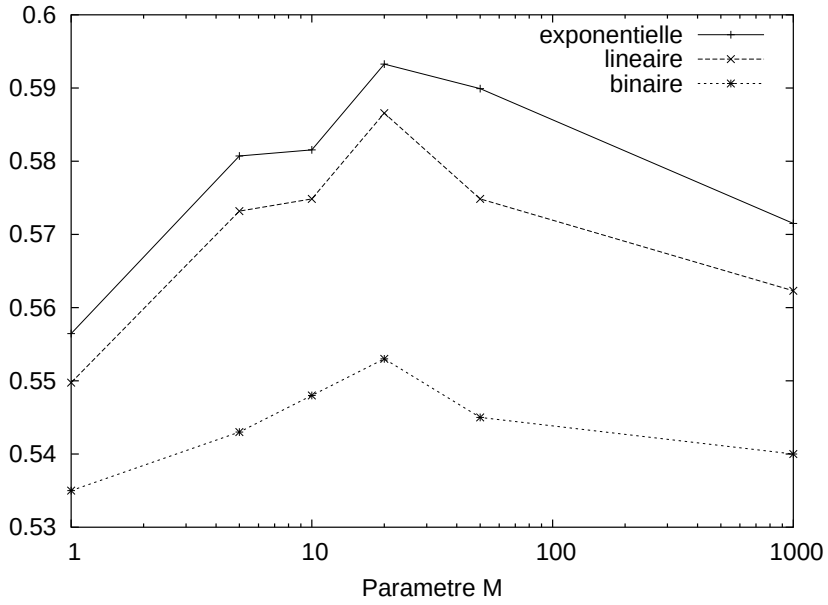


FIG. 6.2 – Performance de RMP avec taux d'activité en fonction du paramètre M et de la fonction f. La valeur $M = \infty$ a été placé à 1000 sur le graphique.

Pour répondre à la première question, nous avons comparé RMP avec taux d'activité ou avec une fenêtre glissante. Comme le corpus Mail Center est de relativement petite taille, il n'est pas possible de mesurer le taux de performance à plusieurs instants. Nous avons donc mesuré le taux de performance sur les 215 derniers documents du corpus et simulé une fenêtre en ne prenant que les x derniers documents comme apprentissage. Le taux d'activité est toujours calculé de façon à ce que le document avec la date moyenne ait un coefficient de $1/2$. La figure 6.3 présente les résultats.

La fenêtre est optimale avec 600 documents (3 lots). Les documents plus anciens correspondent donc majoritairement à des concepts qui n'existent plus. Lorsque la taille de la fenêtre augmente, les performances chutent alors de façon importante. Ce n'est pas le cas avec le taux d'activité avec qui les performances restent stables (ou même augmentent) lorsque la taille de la fenêtre augmente. Il parvient à limiter l'influence des anciens documents et ainsi conserver un taux optimal malgré les nombreux documents inutiles. Il permet donc d'éviter d'avoir à régler attentivement la taille de la fenêtre (qui peut de plus être variable dans le temps), puisqu'il suffit de prendre la fenêtre la plus grande possible et de laisser l'algorithme supprimer de lui-même les documents inutiles. En revanche, nous espérions également que la possibilité de conserver les anciens documents dont le concept est toujours valide permettrait d'améliorer les performances puisque cela permet de garder plus de documents. Cela ne semble malheureusement pas être le cas ici (il y a moins de 0.5% de différence entre le meilleur taux de RMP avec taux d'activité ou avec fenêtre simple). Cette semi-déception peut être dûe à la

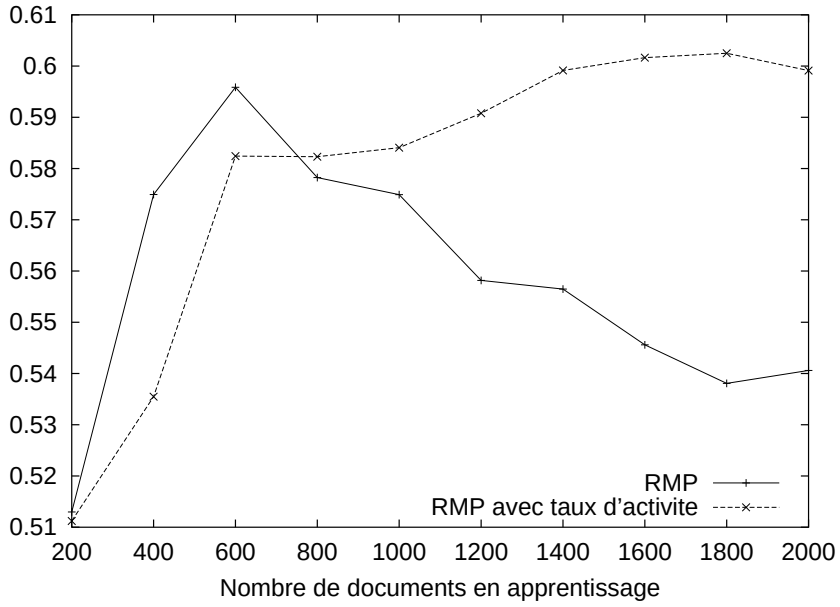


FIG. 6.3 – Taux de performance sur le corpus Mail Center

vitesse de convergence de Rocchio : comme l'algorithme obtient un taux de performance maximum avec peu de documents, il n'a peut-être pas besoin de plus de documents. La réponse à la première question semble donc malgré cette déception être positive : le taux d'activité est équivalent à la meilleure fenêtre possible sans avoir à optimiser la taille de la fenêtre en fonction des évènements.

Pour répondre à la deuxième question, nous avons testé les algorithmes RMP, Rocchio et SVM avec ou sans taux d'activité. Pour les deux derniers classifieurs, qui n'ont pas accès aux clusters, le taux d'activité est calculé sur les classes. Pour SVM, c'est la distance entre l'exemple à classer et les différents hyperplans qui est multipliée par le taux d'activité. Les résultats sont présentés dans le tableau 6.1.

Fonction f	**Rocchio**	**RMP**	**SVM**
Pas de taux d'activité	0.515	0.534	0.584
binaire	0.510/-1%	0.541/1.3%	0.592/1.4%
linéaire	0.548/6.4%	0.591/10.7%	0.606/3.8%
exponentielle	0.555/7.8%	0.594/11.2%	0.606/3.8%

TAB. 6.1 – Performance avec ou sans taux d'activité.

Dans tous les cas, les classifieurs avec taux d'activité obtiennent de meilleures performances que sans. L'amélioration est néanmoins moins sensible pour SVM que pour Rocchio. Ceci peut s'expliquer par le fait que le taux d'activité est employé une fois l'optimisation du classifieur terminée. Il serait préférable d'intégrer le taux directement

pendant l'apprentissage pour lui permettre d'utiliser l'information de manière optimale. Il est probable que l'amélioration de SVM sera plus importante dans ce cas. La différence d'amélioration entre RMP et Rocchio (+11.2% au lieu de +7.8%) provient de la capacité de RMP à sélectionner non pas les classes mais précisément les clusters anciens. La réponse à la deuxième question est donc que le taux d'activité peut être utilisé directement sur les classes mais que les sous-classes apportent des informations supplémentaires réellement utiles.

6.4.2 Sur corpus artificiel

Pour répondre à la troisième question concernant les types de changements que le taux d'activité peut détecter, il nous faut utiliser un corpus artificiel sur lequel nous pouvons créer des changements de concept en spécifiant leurs caractéristiques. La méthodologie est similaire à celle de toutes les expérimentations des travaux présentés dans la section 6.2. Un corpus réel sans changement de concept est découpé en n groupes de documents (appelés *batchs*). Les documents de chaque batch sont classés comme s'ils étaient de nouveaux textes du flux de documents. La taille des batchs est un compromis entre la fiabilité du taux de performance (qui nécessite de tester un grand nombre d'exemples) et la possibilité de tester une évolution temporelle complexe (qui nécessite un nombre important de batchs, chacun d'entre eux sera donc plus petit puisque le corpus total est de taille fixe). Un changement de concept est simulé en changeant la classe des exemples à un certain instant. Ainsi les exemples qui appartenaient à la classe 1 appartiennent ensuite à la classe 2. Le concept de la classe 1 a donc disparu alors qu'un nouveau concept est apparu dans la classe 2.

Pour notre expérimentation, nous avons utilisé le corpus 20 Newsgroups. Ce corpus correspond à un mois de données. Nous supposons ici que les concepts sous-jacents n'ont pas changé au cours de ce mois. Le corpus est découpé aléatoirement en vingt sous-ensembles de 1000 documents chacun. Nous avons réutilisé le corpus hétérogène présenté au chapitre 4. Dans ce corpus, chaque classe est constituée de cinq groupes de discussion de thèmes très différents. La principale nouveauté, que nous souhaitons tester ici, est que lorsque les changements ne s'opèrent pas sur toute la classe mais sur quelques concepts seulement, RMP avec taux d'activité est capable de sélectionner les documents à supprimer. Le tableau 6.2 présente les changements que nous avons construits.

Généralement, les expérimentations sont effectuées sur deux modèles de changements. Dans le *changement brusque*, tous les exemples changent de concept à une date précise. Dans le *changement lent*, les exemples changent peu à peu. Pendant la période de transition, il existe des exemples qui proviennent à la fois de l'ancien et du nouveau concept. Nous avons ajouté un troisième modèle. Dans le *changement en chaîne*, plusieurs changements de concept interviennent dans un court laps de temps. Ce problème modélise

les corpus très dynamiques dans lesquels beaucoup de concepts changent régulièrement. Les trois types de problèmes tels que nous les avons construits sont présentés dans le tableau 6.3.

Pour chaque expérience, nous avons d'abord testé Rocchio avec une fenêtre glissante de 1, 3 ou 5 batchs, afin de visualiser rapidement les comportements les plus simples sur ces trois différentes tâches (voir figure 6.4). Nous avons ensuite comparé RMP avec ou sans taux d'activité sur une fenêtre de 5 batchs. Nous avons utilisé la fonction exponentielle avec $M = 50$. Enfin, nous avons comparé RMP avec taux d'activité et avec fenêtre adaptative suivant la méthode de Klinkenberg [KR98] (voir figure 6.5).

Les résultats de Rocchio avec une fenêtre glissante sont similaires sur les deux premiers corpus. En régime stationnaire, la fenêtre de 1 batch est inférieure aux deux autres. Il y a en revanche très peu de différence entre les fenêtres de 3 et 5 batchs. Rocchio atteint donc son taux optimal avec 3 batchs (c'est-à-dire avec 3000 documents d'apprentissage) et il ne sert à rien d'ajouter plus d'exemples. En période de changements, plus la fenêtre est petite, plus Rocchio est réactif et apprend rapidement les nouveaux concepts. Avec le troisième corpus, les résultats sont beaucoup plus erratiques. Les changements de concept interviennent en moyenne tous les 3 ou 4 batchs. Les fenêtres de 3 ou 5 batchs contiennent donc plus souvent des données qui correspondent à d'anciens concepts que l'inverse. Leur taux de performance n'est donc pas meilleur que si l'on n'utilise qu'un seul batch, malgré le nombre plus important de documents.

L'utilisation du taux d'activité parvient à améliorer sensiblement la réactivité de RMP lors des changements de concept (qu'ils soient brusques ou lents). Pendant la transition du changement lent, il existe des documents dans les deux concepts. Seul le nombre de documents dans chaque concept peut alors être utilisé pour calculer le taux d'activité des deux concepts. L'amélioration est donc plus faible sur le deuxième corpus mais reste néanmoins sensible. Sur le troisième corpus, malgré la difficulté de la tâche le taux d'activité parvient parfaitement à sélectionner les concepts qui ont changé.

Sur les deux premiers corpus, la fenêtre adaptative de Klinkenberg se comporte comme prévue. La fenêtre rétrécit à 1 dès le début du changement de concept brusque. Lorsque le changement est lent, la fenêtre se raccourcit peu à peu jusqu'à atteindre 1 lorsque plus de 50% des documents appartiennent au nouveau concept et elle réaugmente après ce point. Pour ces deux corpus, utiliser une fenêtre adaptative permet d'obtenir une réactivité aux changements légèrement meilleure que le taux d'activité. En revanche, pour le troisième corpus, les changements sont trop fréquents, les taux de performance trop variables et l'algorithme de Klinkenberg ne parvient pas à détecter de changement de concepts (la fenêtre reste à 5 batchs tout le long de l'expérience). Cette méthode est donc inutilisable dans le cas de changements fréquents, alors que le taux d'activité parvient à gérer correctement ce type de changements.

1. comp.graphics	11. alt.atheism	**Avant changement**
2. comp.windows.x	12. sci.electronics	C1 : 1,6,2,12,16
3. comp-os.ms-windows.misc	13. sci.crypt	C2 : 3,7,8,13,17
4. comp.sys.mac.hardware	14. sci.space	C3 : 4,9,10,14,18
5. comp.sys.ibm.pc.hardware	15. sci.med	C4 : 5,11,15,19,20
6. talk.politics.guns	16. misc.forsale	**Après changement**
7. talk.politics.mideast	17. rec.sport.baseball	C1 : **5,11**,2,12,16
8. talk.politics.misc	18. rec.sport.hockey	C2 : **1,6**,8,13,17
9. talk.religion.misc	19. rec.autos	C3 : **3,7**,10,14,18
10. soc.religion.christian	20. rec.motorcycles	C4 : **4,9**,15,19,20

Ch1	Ch2	Ch3	Ch4
1,6 : C1->C2	3,7 : C2->C3	4,9 : C3->C4	5,11 : C4->C1

TAB. 6.2 – Description des changements de concepts construit sur le corpus 20 Newsgroups. Par exemple, pour le changement 1, les exemples des groupes de discussion 1 et 6 appartiennent à la classe C1 avant le changement et font partie de la classe C2 après.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Ch1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
Ch2	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
Ch3	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
Ch4	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

Expérience 1 : avec changement brusque.

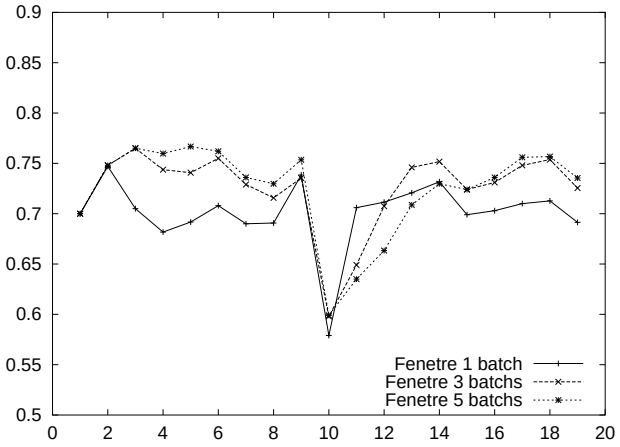
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Ch1	0	0	0	0	0	0	0	0	0.2	0.4	0.6	0.8	1	1	1	1	1	1	1
Ch2	0	0	0	0	0	0	0	0	0.2	0.4	0.6	0.8	1	1	1	1	1	1	1
Ch3	0	0	0	0	0	0	0	0	0.2	0.4	0.6	0.8	1	1	1	1	1	1	1
Ch4	0	0	0	0	0	0	0	0	0.4	0.4	0.6	0.8	1	1	1	1	1	1	1

Expérience 2 : avec changement lent.

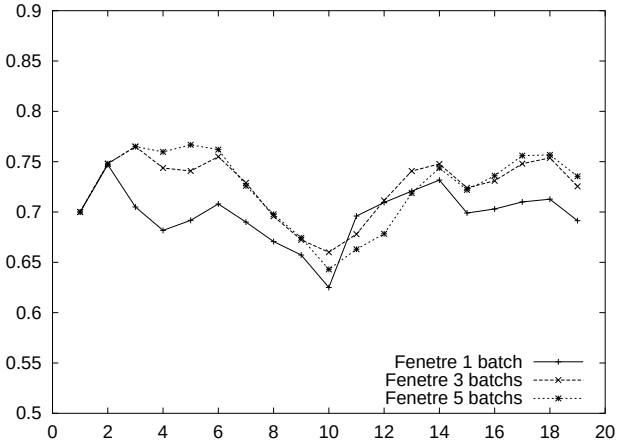
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Ch1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Ch2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
Ch3	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Ch4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Expérience 3 : avec changements en chaîne.

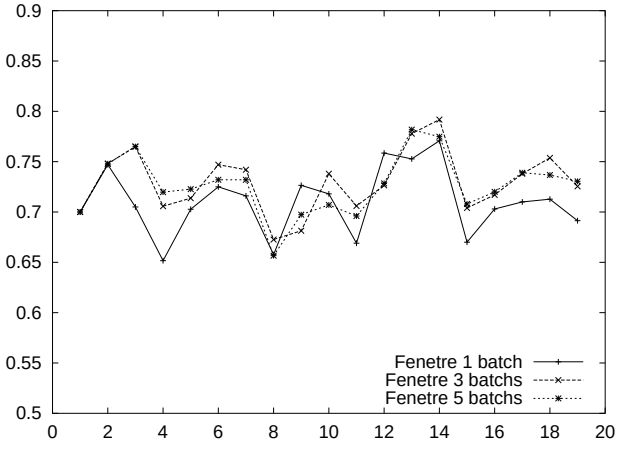
TAB. 6.3 – Description des changements programmés dans les trois expériences. Les valeurs dans les tableaux indiquent la probabilité qu'un exemple soit étiquetté dans le nouveau concept.



Changement brusque



Changement lent



Changements en chaîne

FIG. 6.4 – Comportement de Rocchio en fonction de la taille de la fenêtre pour les trois types de changements.

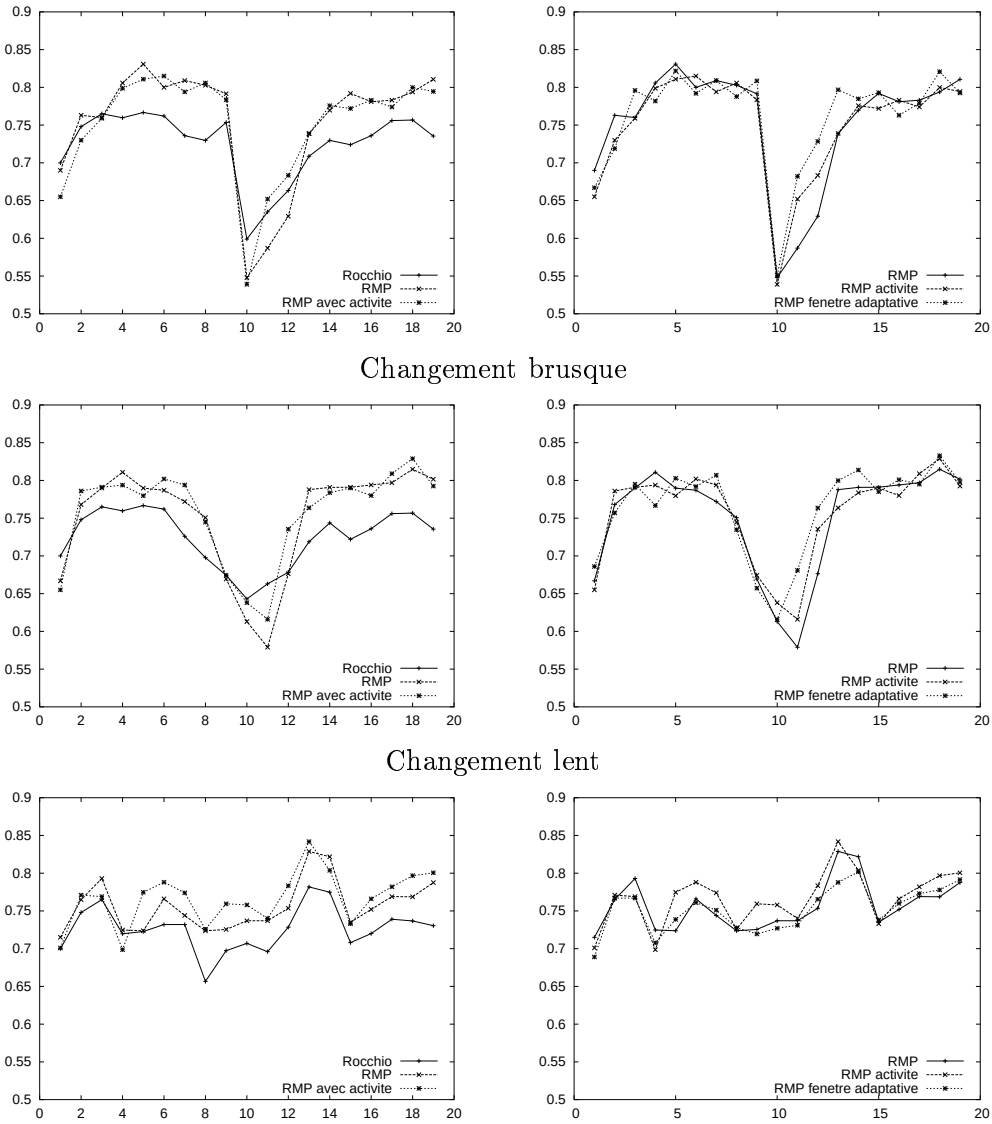


FIG. 6.5 – Comparaison entre Rocchio, RMP, RMP avec taux d'activité et RMP avec fenêtre adaptative.

La réponse à la troisième question est donc que la fenêtre adaptative de type Klinkenberg est préférable dans le cas de changements isolés, mais qu'elle est inopérante dans le cas de changements en chaîne alors que le taux d'activité continue à fonctionner correctement. Cette configuration risque pourtant d'apparaître dans les applications récentes comme le Mail Center.

6.5 Conclusion

Nous avons proposé au chapitre précédent un algorithme de clustering dont le but est d'améliorer les performances de Rocchio dans le cas où les classes ne sont pas thématiques. Ce clustering permet d'obtenir des informations supplémentaires sur le corpus en regroupant les documents sous un deuxième niveau hiérarchique qui correspond à des sous-classes. Dans ce chapitre, nous avons montré comment cette information peut être utilisée pour faire de la détection de changements de concepts sur des corpus temporels.

Cette détection est effectuée à partir de trois types d'indicateurs : les mesures de performance, les propriétés du classifieur et la description des données. La plupart des travaux utilisent les deux premiers indicateurs. Notre méthode s'appuie uniquement sur le troisième type d'indicateur, qui a été peu utilisé jusqu'à présent. L'algorithme calcule un taux d'activité pour chaque cluster, qui joue le même rôle qu'une probabilité d'apparition d'un document à proximité du cluster dans le prochain lot. Ce taux est utilisé comme une probabilité a priori pour baisser le poids des clusters qui ont un faible taux d'activité. La principale innovation vient du fait que nous ne cherchons pas simplement à détecter un changement de concept pour réduire la fenêtre au maximum mais à déterminer quel concept a changé de façon à supprimer les documents qui se rapportent à cet ancien concept. Ceci ne peut se faire que grâce à l'information apportée par le clustering.

Afin de valider notre approche, nous avons testé notre algorithme sur le corpus Mail Center. C'est, à notre connaissance, la première fois qu'un algorithme de détection de changement de concept est testé sur un corpus réel. Les résultats indiquent que le taux d'activité fonctionne correctement. Il permet de conserver une fenêtre de grande taille sans se soucier des changements de concepts et d'obtenir des performances optimales dans tous les cas. Nous avons également comparé notre méthode avec l'algorithme de Klinkenberg de fenêtre adaptative sur un corpus artificiel tiré de 20 Newsgroups. Le système de fenêtre adaptative semble plus réactif aux changements dans le cas de changements isolés. En revanche, cette méthode est inutilisable dans le cas de nombreux changements en chaîne alors que le taux d'activité continue à fonctionner correctement. Ce type de situations risque d'apparaître de plus en plus souvent dans les nouvelles applications de CT, c'est déjà le cas par exemple dans le Mail Center.

Chapitre 7

Conclusions et perspectives

Bilan

Nous nous sommes intéressés au cours de cette thèse aux problèmes soulevés par l'intégration de systèmes de Catégorisation de Textes (CT) dans des applications opérationnelles qui se caractérisent en particulier par des catégories non thématiques. La CT s'était jusqu'à présent limitée à traiter des problèmes de classement de documents suivant leur sujet principal. Cette classification permet de structurer l'accès aux ouvrages d'une bibliothèque en rangeant les documents par thème. Il existe, de nos jours, une demande importante de la part des industriels pour l'intégration de cette technologie dans de nouvelles applications où les catégories ne correspondent plus au sujet principal des documents.

Tout au long de cette thèse, nous avons défendu l'idée que les outils classiques de CT (c'est-à-dire les techniques de représentations des textes et les algorithmes de classification) sont à même de répondre à cette demande.

Nous avons présenté au troisième chapitre quatre nouvelles applications pour la CT qui ont toutes la particularité de chercher à classer les documents dans des catégories qui ne sont pas définies par une thématique précise. Malgré cette caractéristique, les algorithmes de CT peuvent être parfaitement adaptés à ces problèmes. C'est le cas pour la détection de propos raciste comme nous l'avons étudié au cours du projet Princip ou le classement de dépêches suivant l'évènement générateur comme proposé par le projet TDT.

Ces nouvelles applications impliquent souvent des caractéristiques particulières aux-

quelles les techniques de CT doivent s'adapter. Nous avons montré dans le quatrième chapitre quatre exemples de techniques dont l'utilité dépend du contexte applicatif. Ainsi, si la racinisation est habituellement considérée comme plutôt efficace en RI, nous avons montré qu'elle peut avoir un effet désastreux sur les performances en CT. Le Query Zoning est très efficace en filtrage. Il est aujourd'hui considéré comme un moyen simple pour améliorer Rocchio. Pourtant, nous avons montré qu'il n'améliore pas les performances en routage et qu'il est donc complètement inutile dans une application avec ce mode de classification.

Jusqu'à présent, toutes les mesures de performance qui sont utilisées considèrent que les applications utilisent la CT en mode automatique. Dans ce cadre, seule la première proposition de l'algorithme est utilisée et le fait que la réponse correcte soit en seconde position ou à la dernière importe peu. Lorsque les mesures sont généralisées pour refléter un mode d'utilisation plus proche de la réalité qui tient compte des n classes les plus probables, SVM, à l'heure actuelle l'algorithme le plus performant lorsque la mesure est classique obtient des résultats inférieurs aux autres algorithmes et Rocchio devient le plus performant.

Enfin, dans le cas de corpus très bruité (avec beaucoup de documents dont l'étiquette de classe est incorrecte), Rocchio semble très robuste alors que SVM est au contraire très sensible aux erreurs d'étiquetage.

Au cours du chapitre cinq, nous avons étudié en détail le problème des classes multi-thématiques. Pour le résoudre, nous proposons d'utiliser un algorithme de clustering qui détermine de manière faiblement supervisée les sous-thèmes de chaque catégorie. Les sous-classes servent ensuite à modéliser les classes : plus elles sont nombreuses, plus le modèle des classes est complexe. Le clustering est donc un moyen de déterminer la meilleure complexité. Les classifieurs apprennent ensuite les poids optimaux pour un modèle fixé. Nous avons montré que la détection de sous-classes n'est utile que dans le cas où les classes de départ sont thématiquement très hétérogènes. Les algorithmes discriminants ne semblent pas gênés par le manque d'homogénéité dans les classes. Le clustering n'est donc utile que pour les algorithmes génératifs qui ne sont pas capables de modéliser directement des classes complexes.

Nous avons développé un algorithme de clustering qui permet de détecter des clusters lorsque les classes sont hétérogènes et qui conserve les classes d'origine dans le cas contraire. Cet algorithme est utilisé en combinaison avec Rocchio qui est l'un des classifieurs génératifs le plus performant à l'heure actuelle.

Les expériences réalisées montrent que le clustering parvient effectivement à différencier les classes hétérogènes des classes homogènes. Dans ce cas-là, RMP (Rocchio Multi-Prototypes) est effectivement plus performant que Rocchio. Dans le cas contraire, les différences sont quasiment inexistantes. RMP permet surtout d'améliorer la mesure

perf-1 tout en conservant les scores élevés de Rocchio sur perf-x. Il parvient ainsi à se rapprocher sensiblement des résultats de SVM sur perf-1 tout en restant très largement supérieur à SVM avec perf-x. Nous avons montré que l'algorithme CD parvient à sélectionner correctement le nombre optimal de clusters. Par contre, une fois le nombre correct de clusters obtenu, il ne semble pas plus performant que n'importe quel autre algorithme de clustering. Cela signifie qu'il doit être possible d'améliorer RMP en utilisant un algorithme de clustering plus évolué que k-Means tout en conservant ce critère pour calculer le nombre correct de clusters. Enfin, pour que l'utilisation de sous-classes soit réellement intéressante, il est nécessaire d'utiliser avec un corpus de taille importante.

Dans le chapitre six, nous avons montré comment le clustering, qui fournit des informations supplémentaires sur le corpus en regroupant les documents sous un deuxième niveau hiérarchique, peut être utilisée pour faire de la détection de changements de concepts sur des corpus temporels.

Cette détection est effectuée à partir de trois types d'indicateurs : les mesures de performance, les propriétés du classifieur et la description des données. La plupart des travaux utilisent les deux premiers indicateurs. Notre méthode s'appuie uniquement sur le troisième type d'indicateur, qui a été peu utilisé jusqu'à présent. L'algorithme calcule un taux d'activité pour chaque cluster, qui joue le même rôle qu'une probabilité d'apparition d'un document à proximité du cluster dans le prochain lot. Ce taux est utilisé comme une probabilité a priori pour baisser le poids des clusters qui ont un faible taux d'activité. La principale innovation vient du fait que nous ne cherchons pas simplement à détecter un changement de concept pour réduire la fenêtre au maximum mais à déterminer quel concept a changé de façon à supprimer les documents qui se rapportent à cet ancien concept. Ceci ne peut se faire que grâce à l'information apportée par le clustering.

Afin de valider notre approche, nous avons testé notre algorithme sur le corpus Mail Center. C'est, à notre connaissance, la première fois qu'un algorithme de détection de changement de concept est testé sur un corpus réel. Les résultats indiquent que le taux d'activité fonctionne correctement. Il permet de conserver une fenêtre de grande taille sans se soucier des changements de concepts et d'obtenir des performances optimales dans tous les cas. Nous avons également comparé notre méthode avec l'algorithme de Klinkenberg de fenêtre adaptative sur un corpus artificiel tiré de 20 Newsgroups. Le système de fenêtre adaptative semble plus réactif aux changements dans le cas de changements isolés. En revanche, cette méthode est inutilisable dans le cas de nombreux changements en chaîne alors que le taux d'activité continue à fonctionner correctement. Ce type de situations risque d'apparaître de plus en plus souvent dans les nouvelles applications de CT, c'est déjà le cas par exemple dans le Mail Center.

Les qualités principales de notre travail de thèse sont les suivants :

- La place importante de l'implémentation qui se traduit par l'intégration d'un module de CT dans le logiciel Akio Mail Center[1] mais également par le souci constant d'expérimenter systématiquement toutes les techniques proposées sur le plus grand nombre possible de corpus,
- La mise en évidence de la compétitivité de Rocchio lorsque les applications ne sont pas classiques alors qu'il est traditionnellement considéré comme peu performant,
- La démonstration de l'utilité des sous-classes avec les deux exemples de la classification et de la détection de changements de concept.

Les limites de notre travail

L'algorithme de clustering que nous avons mis en place est, dans l'état actuel, déjà opérationnel. Il reste néanmoins quelques défauts auxquels nous n'avons pas eu le temps de remédier. Nous avons insisté au cours du chapitre 6 sur l'importance d'avoir des algorithmes avec apprentissage incrémental. Cette caractéristique est particulièrement difficile à mettre en œuvre dans le cas du clustering comme nous l'avons vu en raison de la mauvaise qualité du clustering incrémental. Notre propre algorithme n'échappe pas à ce défaut. Nous n'avons, pour le moment, pas encore trouvé de moyen pour le rendre incrémental tout en conservant la qualité des clusters. Même en mettant de côté l'aspect non incrémental de l'algorithme, le processus de clustering est relativement lourd. L'un des avantages de Rocchio par rapport à SVM est le faible temps de calcul de l'apprentissage. Cet avantage est réduit avec l'algorithme de clustering. Enfin, alors que nous avons cherché tout au long de cette thèse à analyser systématiquement pour quelles raisons telle ou telle méthode fonctionnait mieux qu'une autre, nous n'avons pas réussi à déterminer précisément quelles sont les caractéristiques des corpus qui indiquent que les sous-classes sont utiles ou non. Notre objectif initial était d'améliorer les classifieurs dans le cas de catégories non thématiques. Pourtant, nous avons trouvé, au cours de nos expériences, des corpus non thématiques sur lesquels les clusters étaient inutiles et inversement un corpus thématique (Reuters en l'occurrence) pour lequel les clusters sont particulièrement utiles. La question reste donc largement ouverte.

Les perspectives

Comme pour toute thèse, nous n'avons pu exploré toutes les pistes comme nous l'aurions souhaité. Il reste de nombreuses questions ouvertes auxquelles nous n'avons pu trouver de réponse. Il reste donc encore beaucoup de travail à effectuer pour s'affranchir des limitations présentées ci-dessus et continuer ce travail de thèse.

[1] `http://www.akio-software.com`

À court terme, il serait intéressant de continuer le travail réalisé sur notre algorithme de clustering pour supprimer les défauts les plus évidents indiqués à la section précédente. Il est impératif, pour pouvoir employer cette méthode dans un cadre opérationnel, de mettre au point une version incrémentale ou au moins semi-incrémentale (qui se met à jour régulièrement au fur et à mesure des arrivées et effectue un ré-apprentissage complet de temps en temps). Une autre façon pour rendre le clustering incrémental serait de fournir une première partition à partir du corpus entier, d'intégrer ensuite les nouveaux documents avec la méthode incrémentale standard et en même temps faire un contrôle des changements de concept. Lorsqu'un changement est détecté, l'algorithme de clustering est alors relancé sur l'ensemble du corpus une nouvelle fois.

Un autre problème important est la lenteur du processus de clustering. Il est donc important de le rendre plus rapide. Cela peut être réalisé en intégrant le critère permettant de mesurer l'hétérogénéité des sous-classes directement dans l'algorithme de formation des clusters alors qu'aujourd'hui, nous effectuons d'abord le clustering, puis nous testons si celui-ci est acceptable au risque de défaire un grand nombre de fois ce qui a été fait. La manière de réaliser cette intégration reste encore entièrement inconnue. Bien sûr, il faudra s'assurer après toute modification de l'algorithme que les clusters construits sont toujours de bonne qualité.

À plus long terme, il existe de nombreux points de questionnement qui peuvent faire l'objet de nouvelles études. Un point important que nous avons indiqué parmi les limites de notre travail concerne l'analyse des caractéristiques des corpus permettant de spécifier à l'avance si les clusters seront utiles ou pas. Cette étape n'est pas nécessaire puisque l'algorithme est déjà capable de déterminer le nombre optimal de clusters. Néanmoins, il serait plus satisfaisant de trouver un critère décelable à l'avance. De façon plus générale, le travail que nous avons réalisé pour connaître à l'avance, sur un corpus donné, à quelle performance on peut s'attendre de la part des algorithmes de CT, reste encore largement non résolu. Bien que très difficile, il est important de continuer à travailler dessus car c'est l'un des défauts importants de la CT qui limite son utilisation dans des applications opérationnelles.

Au cours de cette thèse, l'algorithme de clustering s'est avéré avantageux uniquement pour Rocchio. Nous aimerions expérimenter avec d'autres algorithmes susceptibles de s'améliorer sensiblement par l'utilisation des sous-classes.

Enfin, nous souhaitons continuer à développer l'utilisation du clustering pour d'autres applications. Nous avons présenté au cours de cette thèse son utilisation pour la classification et pour la détection de changements de concept. Nous aimerions l'expérimenter pour de l'aide à la constitution de corpus. Les clusters peuvent ainsi servir à prévenir les utilisateurs lorsque des catégories ont des thèmes proches ou lorsque de nouveaux thèmes apparaissent. Dès que les catégories ne sont plus triviales, et qu'elles commencent à être

structurées dans une hiérarchie de thèmes, l'administration de ces catégories pour les faire évoluer en fonction des documents qui arrivent devient une tâche complexe et qui demande beaucoup de temps. Elles sont généralement gérées par plusieurs personnes ce qui multiplie les incohérences (des catégories dupliquées à plusieurs endroits de l'arbre par exemple). Le Mail Center et les annuaires en ligne par exemple sont des applications qui pourraient profiter beaucoup de ce genre d'outils.

Annexe

A

Détails sur l'implémentation

A.1 Introduction

Nous avons, tout au long de cette thèse, donné une place importante à l'implémentation. Afin que toute cette partie de notre travail soit également mise en valeur, nous avons décidé de la présenter en annexe.

Le premier travail important a été la réalisation du moteur de catégorisation de textes qui a été intégré dans le logiciel Akio Mail Center[1]. Le moteur principal a été écrit en C++ en collaboration avec Guillaume Schmid, un employé d'Akio Solutions. Ce moteur a ensuite été intégré dans le logiciel via des interfaces Corba pour les phases d'apprentissage hors ligne et via des librairies écrites en Perl pour la phase interactive de classification.

Le moteur avait été, depuis le départ, conçu spécifiquement pour les besoins particuliers du logiciel Akio Mail Center. Il n'est pas adapté aux autres besoins que nous avons pu avoir au cours de la thèse. Afin de pouvoir réaliser toute sorte d'expérimentations, le moteur a été entièrement réécrit de façon plus modulaire. La deuxième version peut désormais être utilisée aussi bien en mode batch pour des expérimentations qu'en mode d'exploitation classique. Le programme final fait environ 10 000 lignes de code en C++.

Le reste des fonctions nécessaires à l'automatisation des tests ont été réalisées en Perl. Cela va donc de la gestion des corpus pour les faire passer du format de départ (XML ou autre) à notre format spécifique, à la gestion des tests avec validation croisée et calcul des tests statistiques en passant par la transformation des corpus pour les ex-

[1] `http://www.akio-software.com`

périmentations avec corpus artificiellement modifiés. La seule méthode que nous n'ayons pas implémentée nous-même est SVM. Nous utilisons nos méthodes de pré-traitement pour transformer un texte en un vecteur numérique puis l'apprentissage proprement dit est effectué par le logiciel SVM$^{\text{light}}$ de T. Joachims[2]. Nous avons également utilisé les libraires `gmime` en C[3] et MailTools en Perl[4] pour parser le format des emails et `unac`[5] pour désaccentuer les caractères.

A.2 Les différents programmes

Le moteur de catégorisation de textes est constitué essentiellement de quatre fonctions, qui sont concrétisées par quatre programmes distincts : `clusterdoc` gère l'algorithme de clustering RMP, `indexdoc` gère la phase d'apprentissage, `extractcarac` extrait des informations synthétiques sur le corpus une fois l'apprentissage réalisé, `querydoc` gère la phase de test.

`clusterdoc` prend en entrée un corpus étiqueté de documents et fournit la liste des clusters à créer. Toutes les méthodes que nous avons présentées au cours de cette thèse (k-Means, clustering hiérarchique descendant greedy, avec critère PRC et CD) sont utilisables avec leurs différents paramètres en spécifiant une option sur la ligne de commande.

`indexdoc` permet d'indexer un corpus suivant la méthode Rocchio. Le corpus est ici sous la forme d'une hiérarchie à trois niveaux : documents / clusters / classes. `indexdoc` calcule un profil pour chaque cluster et sauvegarde les informations sous la forme d'un index inversé (on liste pour chaque mot, l'ensemble des clusters qui le contient).

Notre programme gère donc nativement RMP et considère Rocchio et les k-PPV comme des cas particuliers de RMP. Pour Rocchio, il suffit de construire un cluster par classe. Pour k-PPV, il faut construire un cluster par document (la méthode par défaut permet ainsi de faire du 1-PPV ; la gestion du k-PPV avec $k \neq 1$ doit ensuite être prise en compte lors des tests).

`extractcarac` permet essentiellement de récupérer les informations sur les profils de chaque cluster. Ces informations étant sauvegardées sous un format binaire, ce programme permet de les afficher dans un format plus facilement exploitable par un humain. Les informations retournées sont les n mots les plus importants de chaque cluster, les normes de chacun d'entre eux ou la liste des mots avec leur IDF associé.

[2] `http://svmlight.joachims.org`
[3] `http://spruce.sourceforge.net/gmime/`
[4] `http://search.cpan.org/dist/MailTools/`
[5] `http://www.nongnu.org/unac/`

`querydoc` s'occupe de la phase de test. À partir des données fournies par `indexdoc`, il prend chaque texte en entrée un à un et fournit la liste des classes les plus pertinentes. Par défaut, la méthode utilisée est "1 plus proche cluster" ce qui correspond à la stratégie de Rocchio et de RMP. Pour k-PPV, le programme peut être configuré pour utiliser la méthode "k plus proches clusters" avec un vote pondéré par la similarité ou non.

A.3 Configurabilité du programme

Le but principal du développement de la seconde version du logiciel était d'obtenir un programme très configurable afin de pouvoir avoir un programme à la fois efficace en mode d'exploitation et de test.

Nous avons utilisé énormément les designs patterns "Stratégie" et "Patron de méthode" qui sont des formes de design recommandées lorsqu'une application doit gérer plusieurs méthodes concurrentes pour une même tâche.

A.3.1 Les Pré-traitements

Afin de tester plusieurs pré-traitements différents, nous avons une classe générique dont l'interface est minimale : elle ne contient qu'une fonction `transform` qui prend un texte en entrée et fournit un texte transformé en sortie. Cette interface est ensuite spécialisée en deux classes. La première s'occupe des transformations globales à opérer sur un document entier. Par exemple un filtre HTML qui supprime toutes les informations de formatage HTML pour ne conserver que le texte est une opération globale. La deuxième classe s'occupe des transformations locales à opérer sur un mot. L'opération de racinisation est typiquement une opération locale.

Cette organisation est typiquement celle du design pattern "Stratégie". Elle est présentée dans la figure A.1 suivant un diagramme de type UML. Les classes abstraites `LocalTr` et `GlobalTr` sont des interfaces. Seules les classes filles implémentent la méthode `transform` suivant une stratégie donnée. Parallèllement, la classe `Lexer` est de type "Patron de méthode". Avant l'indexation du corpus, elle enregistre l'ensemble des transformations, locales ou globales, qui seront utilisées. À l'indexation, elle prend chaque texte, appelle les différentes transformations globales, segmente le texte en mots et appelle les différentes transformations locales avant de stocker chaque mot dans le `Dico` et la structure `DocTF`. L'aspect "Patron de méthode" apparaît dans le sens où l'algorithme contient des zones non spécifiées sous la forme d'une liste de transformations à appliquer.

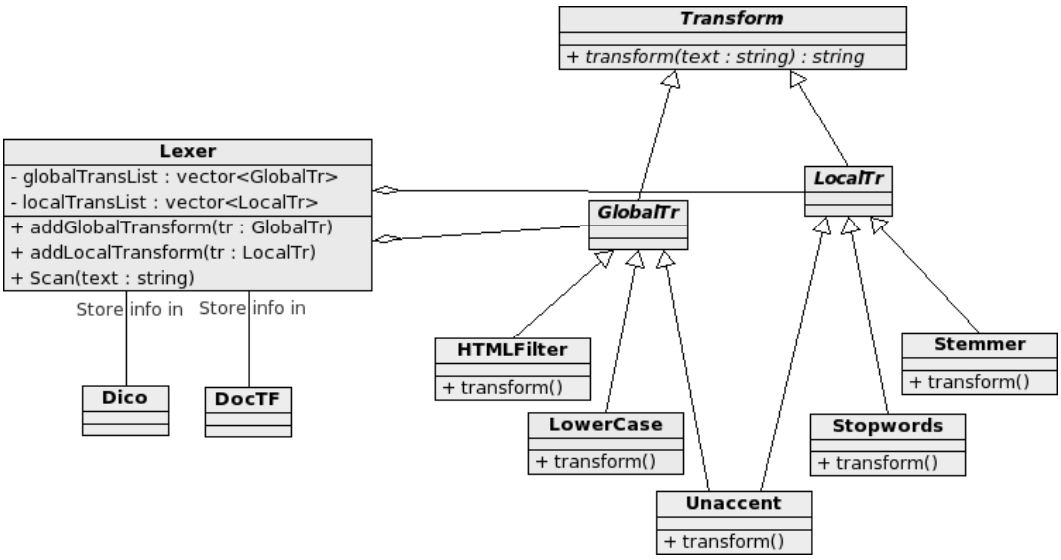


FIG. A.1 – Diagramme UML du pattern Stratégie pour les pré-traitements.

A.3.2 Matrice d'occurrence en RAM ou sur disque

Un logiciel de catégorisation de textes doit pouvoir traiter un très grand nombre de documents. Les données à conserver (les occurrences de chaque mot dans chaque document) sont très nombreuses. Une question importante qui se pose est : faut-il conserver ces informations en RAM afin que le processus d'indexation soit le plus rapide possible ou les mettre sur disque quitte à se retrouver limiter par la faible vitesse des entrées-sorties. Clairement, en phase d'exploitation, il est préférable de mettre toutes les informations sur disque car (i) on ne sait pas a priori quelle est la taille du corpus (il est donc impossible de savoir combien de RAM le programme va utiliser), (ii) la phase d'apprentissage est généralement réalisée hors ligne, le temps n'est alors pas un facteur limitatif. Dans le cas d'expérimentations sur corpus, nous savons sur quel corpus nous travaillons, et comme les expérimentations concernent essentiellement les méthodes d'apprentissage, il est important d'être rapide pendant cette phase puisque nous allons la lancer beaucoup de fois. Dans ce cadre, il est alors préférable, quand c'est possible (c'est-à-dire si le corpus n'est pas trop volumineux), de mettre toutes les informations en RAM.

Le choix du stockage en RAM ou sur disque est réalisé à la compilation. Nous n'avons pas eu l'utilité de le rendre dynamique mais il est tout à fait possible de rendre le système modifiable à la volée une fois encore avec le pattern "Stratégie" (par exemple en basculant d'une méthode à l'autre en fonction de la taille du corpus).

Pour donner un ordre d'idée, pour indexer 200 000 documents du corpus Reuters (sans faire la partie clustering de RMP mais uniquement le calcul des prototypes Rocchio),

il faut 45 minutes (avec un CPU utilisé en moyenne à 25%) et 360 Mo de RAM avec la matrice en RAM, 5 heures (avec un CPU utilisé à 5%) et 80 Mo de RAM. Dans les deux cas, le faible taux d'utilisation du CPU indique que le processus passe beaucoup de temps en attente d'entrée/sortie sur disque (pour récupérer les documents et stocker les informations finales).

A.3.3 Autres exemples de configurabilité

Le pattern "Stratégie" est également utilisé pour choisir comment la liste des documents du corpus est récupérée. La classe `Listdoc` contient la structure en RAM et toutes les méthodes pour accéder à la liste depuis les autrs classes. La méthode `createList` est ensuite implémentée par les classes filles `FicListDoc` et `DBListDoc` qui récupèrent les informations respectivement depuis une liste dans un fichier et depuis une base de données.

Le même pattern est encore utilisé pour l'implémentation des différentes méthodes d'apprentissage (Rocchio avec ou sans éléments négatifs, avec ou sans Query Zoning, clustering avec quel critère, etc.)

A.3.4 Algorithme incrémental

Nous n'avons pas implémenté d'algorithmes incrémentaux pour faire l'apprentissage comme nous en parlons dans le chapitre 6. En effet, dans les expérimentations que nous avons réalisées, les corpus sont systématiquement statiques. Même dans les expériences du chapitre 6, les tests étant réalisés par batch, l'incrémentalité est peu utile dans ce cas. Néanmoins, cela ne poserait pas de problème particulier pour intégrer ce type de possibillité.

A.4 Description détaillée

Les méthodes d'extraction d'informations depuis les structures `InvIndex` et `DocTF` ainsi que les méthodes de calcul de similarité lors des phases de test n'étant pas très intéressantes, nous ne décrivons que les parties indexation des documents et algorithme de clustering.

A.4.1 Indexation des documents

Le coeur du moteur d'indexation est découpé en deux parties bien distinctes. La première gère toutes les structures de données et la deuxième s'occupe des algorithmes eux-mêmes. Afin de rendre les différents composants les plus réutilisables possibles, les deux types de classe sont en contact uniquement via leurs interfaces. Seule la classe Dico n'a pas d'interface. Les données qu'elle gère sont utilisées très régulièrement et

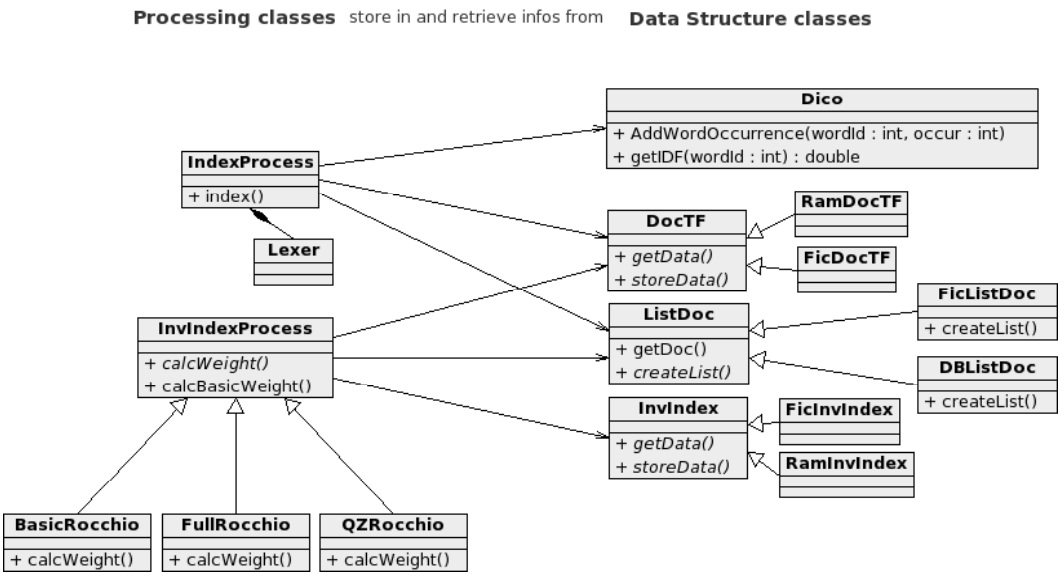


FIG. A.2 – Diagramme UML du coeur du moteur d'indexation.

sont relativement peu nombreuses. La meilleure solution est toujours de les conserver en RAM et il n'y a donc pas de raison de créer une interface, qui complexifie l'application pour rien. Pour les données importantes `DocTF` et `InvIndex`, il existe deux classes qui implémentent chaque interface : une qui stocke toutes les informations en RAM et une autre sur disque. `ListDoc` existe également en deux variations suivant que la liste des documents est récupérée via un fichier ou via une base de données. De la même façon, les méthodes de calcul ont été virtualisées via des interfaces. Le calcul des profils de chaque prototype est donc géré par l'interface `InvIndexProcess` qui est implémentée indifféremment par les classes `BasicRocchio`, `FullRocchio` (Rocchio avec éléments négatifs), ou `QZRocchio` (Rocchio avec Query Zoning).

Le processus complet commence donc par donner la main à `IndexProcess` qui prend la liste des documents de l'objet `ListDoc` et demande l'indexation de chaque document à la classe `Lexer`. Celle-ci indexe l'ensemble des mots en inscrivant les informations dans le `Dico` pour l'IDF et dans le `DocTF` pour les TF. Une fois tous les documents indexés, une deuxième passe de `IndexProcess` permet de déterminer les pondérations de chaque terme pour chaque document (avec le TF, l'IDF et la normalisation). Le processus passe alors à `InvIndexProcess` qui passe en revue une nouvelle fois le `ListDoc` pour relire les `DocTF` et construire itérativement le profil de chaque cluster en les stockant dans le `InvIndex`. Comme pour les `DocTF`, une deuxième passe permet enfin de calculer les normes de chaque cluster.

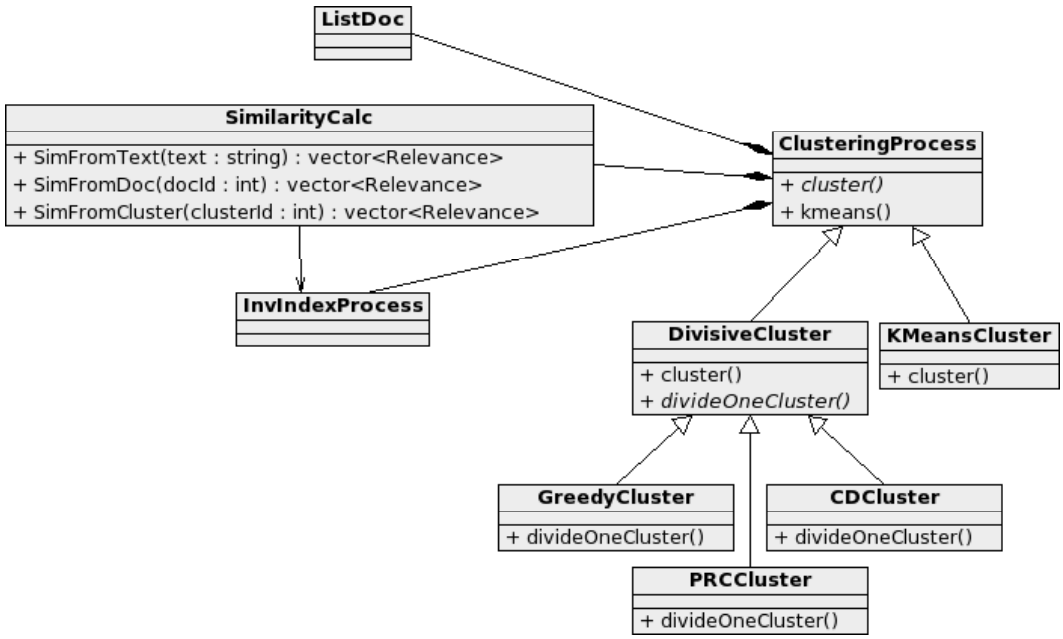


FIG. A.3 – Diagramme UML des classes de clustering.

A.4.2 Algorithme de clustering

Le processus de clustering nécessite la même architecture que pour l'indexation puisqu'il faut également construire les profils de chaque cluster pour mesurer des similarités. Nous ne détaillons pas cette partie du diagramme qui est identique à la section précédente. Le processus est essentiellement formé de deux classes. `ClusteringProcess` s'occupe de l'algorithme de clustering proprement dit. `SimilarityCalc` s'occupe de mesurer les valeurs de similarité entre deux documents, entre un document et un cluster et entre deux clusters. Cette classe est également utilisée lors de la phase de test. La classe de base `ClusteringProcess` contient une méthode abstraite `cluster()` qui est réimplémentée par les classes filles. Cette méthode exécute le clustering et fournit un `ListDoc` mis à jour en fonction des nouveaux clusters. Tous les algorithmes de clustering utilisant à un moment l'algorithme k-Means, celui-ci est implémenté dans la classe de base via la méthode `kmeans`. La classe `DivisiveCluster`, qui hérite de `ClusteringProcess` est encore une classe abstraite. Elle définit le principe général de clustering hiérarchique descendant en appelant à chaque étape la méthode `DivideOneCluster` qui est, elle, abstraite et définie dans les sous-classes `Greedy`, `PRC` et `CD`.

A.4.3 Les scripts Perl

Bien que le langage Perl puisse être utilisé pour faire des scripts à usage unique qui sont vite écrits et vite oubliés, nous avons toujours cherché à fournir des scripts

suffisamment génériques et correctement écrits pour pouvoir les réutiliser. La plupart de nos scripts tiennent néanmoins en un fichier, et il était rarement nécessaire de faire un design objet soigné. Nous nous contenterons ici de lister l'ensemble des fonctions qui ont été réalisées en Perl :

- Passage du format brut des corpus à notre propre format (récupération des données via des fichiers XML ou dans des bases de données),
- Transformation d'un corpus suivant le modèle de tâche (filtrage ou routage, 1 contre 1 ou 1 contre tous),
- Création des corpus artificiels (corpus bruités et avec changements de concepts),
- Calcul du taux d'activité de chaque cluster dans le cas des corpus temporels,
- Mise en place des protocoles de test (validation croisée, test de McNemar, t-test),
- Analyse qualitative des résultats (affichage des mots pertinents pour chaque cluster, comparaison de deux clusterings).

Bibliographie

[Aha97] David W. Aha, editor. *Lazy Learning*, volume 11(1-5). Kluwer Academic Publishers, from artificial intelligence review edition, 1997.

[AKCS00] Ion Androutsopoulos, John Koutsias, Konstandinos V. Chandrinos, and Constantine D. Spyropoulos. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 160–167. ACM Press, 2000.

[AL01] Rie Kubota Ando and Lillian Lee. Iterative Residual Rescaling : An analysis and generalization of LSI. In *Proceedings of the 24th SIGIR*, pages 154–162, 2001.

[APL98] James Allan, Ron Papka, and Victor Lavrenko. On-line new event detection and tracking. In *Research and Development in Information Retrieval*, pages 37–45, 1998.

[ARP] ARPA. Message Understanding Conference. `http://cs.nyu.edu/cs/faculty/grishman/muc6.html`.

[AS] ACL-SIGLEX. Senseval, evaluation of word sense disambiguation systems. `http://www.senseval.org`.

[ASS00] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing Multiclass to Binary : a unifying Approach for Margin Classifiers. *Journal of Machine Learning Research*, 1 :113–141, 2000.

[Ber99] Adam Berger. Error-correcting output coding for text classification. In *In IJCAI'99 : Workshop on Machine Learning for Information Filtering*, Sweeden, 1999.

[Bes02] Romaric Besançon. *Intégration de connaissances syntaxiques et sémantiques dans les représentations vectorielles de textes.* PhD thesis, EPFL, Lausanne, 2002.

[BGT03] Ibtissam Bensetti, Guilhem Grosso, and Hind Touach. Étude et analyse des différentes méthodes anti-spam. Master's thesis, ENST, 2003.

[BL97] Rakesh D. Barve and Philip M. Long. On the complexity of learning from drifting distributions. *Information and Computation*, 138(2) :101–123, 1997.

[BM98a] L. Douglas Baker and Andrew K. McCallum. Distributional clustering of words for text classification. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 96–103. ACM Press, 1998.

[BM98b] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[BMP02] R. Basili, A. Moschitti, and M.T. Pazienza. Empirical investigation of fast text categorization over linguistic features. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI 2002)*, France, 2002.

[BMRR02] Daniel Beauchêne, Cécile Million-Rousseau, and Christine Rieu. Détection automatique de l'insatisfaction du client dans un contexte de commerce électronique. In *Actes du Colloque International sur la Fouille de Texte (CIFT) dans le cadre de CFD'02*, pages 108–118, 2002.

[Bre94] Leo Breiman. Bagging predictors. Technical Report 421, University of California, Berkeley, CA, 1994.

[BS95] Chris Buckley and Gerard Salton. Optimization of relevance feedbacks weights. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference of Research and Development in Information Retrieval*, pages 351–357, 1995.

[BSA00] Stephan Busemann, Sven Schmeier, and Roman G. Arens. Message classification in the call center. In *Proc. of the 6th Conference on Applied Natural Language Processing*, pages 159–165, Seattle, WA, 2000.

[BSH00] Armelle Brun, Kamel Smaili, and Jean-Paul Haton. Experiment analysis in newspaper topic detection. *String Processing and Information Retrieval - SPIRE, IEEE Computer Society*, 2000.

[BSQ03] Li Baoli, Yu Shiwen, and Lu Qin. An improved k-nearest neighbor algorithm for text categorization. In *Proceedings of the 20th International Conference on Computer Processing of Oriental Languages*, China, 2003.

[Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2) :121–167, 1998.

[CCC99] Jennifer Chu-Carroll and Bob Carpenter. Vector-based natural language call routing. *Computational Linguistics*, 25(3) :361–388, 1999.

[CCJ+99] Andrew J. Carlson, Chad M. Cumby, Jeff, L. Rosen, and Dan Roth. *SNoW User's Guide*. UIUC, 1999.

[CCSC03] Chien Chin Chen, Yao-Tsung Chen, Yeali Sun, and Meng Chang Cen. Life cycle modeling of news events using aging theory. In *Proceedings of the 14th European Conference on Machine Learning (ECML)*, pages 47–59, Croatia, 2003.

[CDF+98] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 509–516. AAAI Press, 1998.

[CES00] E. Catona, D. Eichmann, and P. Srinivasan. Filters and answers : The University of Iowa TREC-9 results. In *Proceedings of The Ninth Text REtrieval Conference (TREC-9)*, pages 533–542, 2000.

[CG95] Ken Church and Bill Gale. Inverse Document Frequency (IDF) : A measure of deviations from Poisson. In *The Third Workshop on Very Large Corpora (in SIGDAT and SIGNLL)*, 1995.

[CH67] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1) :1179–1184, 1967.

[CKS+88] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. AutoClass : A Bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64, Ann Arbor, June 1988. Morgan Kaufmann Publishers.

[CM01] Xavier Carreras and Lluís Márquez. Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing*, pages 58–64, Bulgaria, 2001.

[CNI02] CNIL. Bilan de l'opération "boîte à spam", 2002. http ://www.cnil.fr/actu/communic/actu45.htm.

[Coh96] William Cohen. Learning rules that classify e-mail. In AAAI Press, editor, *Proceedings of the AAAI Spring Symposium on Machine Learning and Information Access*, pages 18–25, 1996.

[Coo99] Robert Cooley. Classification of news stories using Support Vector Machines. In *IJCAI'99 Workshop on Text Mining*, Stockholm, Sweden, 1999.

[Cox62] David R. Cox. *Renewal theory*. Mathuen, 1962.

[CS96] William W. Cohen and Yoram Singer. Context-sensitive learning methods for text categorization. In *Proceedings of SIGIR-96, 19th ACM*

International Conference on Research and Development in Information Retrieval, pages 307–315, Zürich, CH, 1996. ACM Press.

[CS02] Koby Crammer and Yoram Singer. A new family of online algorithms for category ranking. In *Proceedings of the 25rd Conference on Research and Development in Information Retrieval (SIGIR)*, pages 151–158, Finland, 2002.

[DB95] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2 :263–286, 1995.

[DC96] Harris Drucker and Corinna Cortes. Boosting decision trees. *Advances in Neural Information Processing Systems*, pages 479–485, 1996.

[DDL+90] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6) :391–407, 1990.

[DE95] Ido Dagan and Sean P. Engelson. Committee-based sampling for training probabilistic classifiers. In *International Conference on Machine Learning*, pages 150–157, 1995.

[Die97] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithm. Technical report, Departement of Computer Science, Oregon State University, 1997.

[DK95] Thomas Dietterich and Eun Kong. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, Department of Computer Science, Oregon State University, 1995.

[dKMK96] H. de Kroon, T. Mitchell, and E. Kerckhoffs. Improving learning accuracy in information filtering. In *International Conference on Machine Learning - Workshop on Machine Learning Meets HCI*, 1996.

[DKR97] Ido Dagan, Yale Kerov, and Dan Roth. Mistake-driven learning in text categorization. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 55–63. ACL, 1997.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1) :1–38, 1977.

[DMKS00] Stephen D'Alessio, Keitha Murray, Aaron Kershenbaum, and Robert Schiaffino. The effect of using hierarchical classifiers in text categorization. In *Proceedings of RIAO-2000*, pages 302–313, Paris, 2000.

[DP96] Pedro Domingos and Michael J. Pazzani. Beyond independence : Conditions for the optimality of the simple bayesian classifier. In *International Conference on Machine Learning*, pages 105–112, 1996.

[DPH98] Susan Dumais, John Platt, and David Heckerman. Inductive learning algorithms and representations for text categorization. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 148–155, 1998.

[DWV99] Harris Drucker, Donghui Wu, and Vladimir Vapnik. Support Vector Machines for spam categorization. *IEEE Transactions on Neural networks*, 10(5) :1048–1054, 1999.

[ERS+99] D. Eichmann, M. Ruiz, P. Srinivasan, N. Street, C. Chris, and F. Menczer. A cluster based approach to tracking, detection and segmentation of broadcast news. In *Proceedings of the DARPA Broadcast News Workshop*, pages 69–76, 1999.

[FBF77] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3) :209–226, 1977.

[FH01] Karel Fuka and Rudolf Hanka. Feature set reduction for document classification problems. In *IJCAI-01 Workshop : Text Learning : Beyond Supervision*, 2001.

[FM97] Yoav Freund and Yishay Mansour. Learning under persistent drift. In *Proceedings of the European Conference on Computational Learning Theory*, pages 109–118, 1997.

[Fri96] Jerome H. Friedman. Another approach to polychotomous classification. Technical report, Stanford University, 1996.

[FS97] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1) :119–139, 1997.

[GAA+02] Allen Gorin, Alicia Abella, Tirso Alonso, Giuseppe Riccardi, and Jeremy Wright. Automated natural spoken dialog. *IEEE Computer Magazine*, 35(4) :51–56, 2002.

[GGPC02] Eric Gaussier, Cyril Goutte, Kris Popat, and Francine Chen. A hierarchical model for clustering and categorising documents. In *Proceedings of 24th European Colloquium in IR research (ECIR)*, pages 229–247, 2002.

[Gha01] Rayid Ghani. Using error-correcting codes for efficient text classification with a large number of categories. Master's thesis, Carnegie Mellon University, Center for Automated Learning and Discovery, 2001.

[GHMLPS00] José M. Gómez Hidalgo, Manuel Maña López, and Enrique Puertas Sanz. Combining text and heuristics for cost-sensitive spam filtering. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 99–102. Lisbon, Portugal, 2000.

[GSS00] Luigi Galavotti, Fabrizio Sebastiani, and Maria Simi. Feature selection and negative evidence in automated text categorization. In *Proceedings of the ACM KDD-00 Workshop on Text Mining*, Boston, US, 2000.

[HANS90] P. J. Hayes, P. M. Andersen, I. B. Nirenburg, and L. M. Schmandt. TCS : a shell for content-based test categorization. In *Proceedings of CAIA-90, 6th IEEE Conference of Artificial Intelligence Applications*, pages 320–326, 1990.

[HBLH94] W. Hersh, C. Buckley, T.J. Leone, and D. Hickam. OHSUMED : an interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–201, 1994.

[HK00] Eui-Hong Han and George Karypis. Centroid-based document classification : Analysis and experimental results. In *Principles of Data Mining and Knowledge Discovery*, pages 424–431, 2000.

[HKK01] Eui-Hong Han, George Karypis, and Vipin Kumar. Text categorization using weight adjusted k -nearest neighbor classification. *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2035 :53–65, 2001.

[HL94] David P. Halmbold and Philip M. Long. Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14(1) :27–45, 1994.

[HL02] C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2) :415–425, 2002.

[Hof00] Thomas Hofmann. Probmap - a probabilistic approach for mapping large document collections. *Journal for Intelligent Data Analysis*, 4 :149–164, 2000.

[HPRZ02] Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint Classification : a new Approach to Multiclass Classification and Ranking. In *Proceedings of the 2002 Conference on Advances in Neural Information Processing Systems (NIPS'02)*, 2002.

[HT98] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems (NIPS-10)*, pages 507–513. MIT Press, 1998.

[Hul98] David Hull. The TREC-7 filtering track : Description and analysis. In *The Seventh Text REtrieval Conference*, pages 33–56, 1998.

[HW79] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28 :100–108, 1979.

[IFKM03] Makoto Iwayama, Atsushi Fujii, Noriko Kando, and Yuzo Marukawa. An empirical study on retrieval models for different document genres : patents and newspaper articles. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 251–258. ACM Press, 2003.

[IJ01] Gabriel Illouz and Michèle Jardino. Analyse statistique et géométrique de corpus textuels. *Traitement Automatique des Langues*, 42(2) :501–516, 2001.

[IT95] Makoto Iwayama and Takenobu Tokunaga. Hierarchical Bayesian clustering for automatic text classification. In *Proceedings of IJCAI-95, 14th International Joint Conference on Artificial Intelligence*, pages 1322–1327. Morgan Kaufmann Publishers, 1995.

[JF87] William Jones and George Furnas. Pictures of relevance : A geometric analysis of similarity measures. *Journal of the American Society for Information Science*, 38(6) :420–442, November 1987.

[Joa97] Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of the 14th International Conference on Machine Learning, (ICML)*, pages 143–151, Nashville, US, 1997. Morgan Kaufmann Publishers.

[Joa98] Thorsten Joachims. Text categorization with Support Vector Machines : Learning with many relevant features. In *ECML-98, 10th European Conference on Machine Learning*, pages 137–142, 1998.

[Joa99] Thorsten Joachims. *Advances in Kernel Methods - Support Vector Learning*. B. Schölkopf and C. Burges and A. Smola, MIT-Press, 1999.

[Joa01] Thorsten Joachims. A statistical learning model of text classification with Support Vector Machines. In *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*, pages 128–136, New Orleans, US, 2001. ACM Press.

[JWR00] Karen Sparck Jones, Steve Walker, and Stephen E. Robertson. A probabilistic model of information retrieval : development and comparative ex-

periments - part 2. *Information Processing and Management*, 36(6) :809–840, 2000.

[KA01] Aleksander Kolcz and Joshua Alspector. SVM-based filtering of e-mail spam with content-specific misclassification costs. In *Proceedings of ICDM-2001 Workshop on Text Mining (TextDM 2001)*, 2001.

[Kat96] Slava M. Katz. Distribution of content words and phrases in text and language modelling. *Journal of Natural Language Engineering*, 2(1) :15–59, 1996.

[KH00] George Karypis and Eui-Hong Han. Concept indexing : A fast dimensionality reduction algorithm with applications to document retrieval and categorization. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM)*, 2000.

[KHZ00] Yu-Hwan. Kim, Shang-Yoon Hahn, and Byoung-Tak Zhang. Text filtering by boosting Naive Bayes classifiers. In ACM Press, editor, *Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 168–175, 2000.

[KJ00] Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 487–494. Morgan Kaufmann Publishers, 2000.

[Kos00] Leila Kosseim. Systèmes de réponse automatique : État de l'art. Technical report, Département d'informatique et de recherche opérationnelle, Université de Montréal, 2000.

[KR98] Ralf Klinkenberg and Ingrid Renz. Adaptive information filtering : Learning in the presence of concept drifts. In AAAI Press, editor, *workshop notes of Learning for Text Categorization*, pages 33–40, 1998.

[KS96] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *Proceedings of the International Conference on Machine Learning*, pages 284–292, 1996.

[KS97] Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pages 170–178. Morgan Kaufmann Publishers, 1997.

[Kwo98] James T. Kwok. Automated text categorization using Support Vector Machine. In *Proceedings of ICONIP'98, 5th International Conference on Neural Information Processing*, pages 347–351, Kitakyushu, JP, 1998.

[Lam98] Wai Lam. Using a generalized instance set for automatic text categorization. In *Proceedings of SIGIR-98, 21th ACM International Conference on Research and Development in Information Retrieval*, pages 81–89, 1998.

[Lan95] Ken Lang. Newsweeder : Learning to filter news. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML)*, pages 331–339, 1995.

[Lan99] Carsten Lanquillon. Information filtering in changing domains. In *Proceedings Workshop on Machine Learning for Information Filtering, International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 41–48, Stockholm, Sweden, 1999.

[Lar98] Leah Larkey. Some issues in the automatic classification of U.S. patents. In *Working Notes for the AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[LC90] David D. Lewis and W. Bruce Croft. Term clustering of syntactic phrases. In *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 385–404, 1990.

[LC96] Leah S. Larkey and W. Bruce Croft. Combining classifiers in text categorization. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 289–297, Zürich, CH, 1996. ACM Press.

[Lew92] David D. Lewis. An evaluation of phrasal and clustered representations on a text categorization. In *Proceedings of the 15th ACM International Conference on Research and Development in Information Retrieval*, pages 37–50, Denmark, 1992.

[Lew98] David D. Lewis. Naive (Bayes) at forty : The independence assumption in Information Retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 4–15, 1998.

[LJ98] Y. H. Li and A. K. Jain. Classification of text documents. *The Computer Journal*, 41(8), 1998.

[LK02] Edda Leopold and Jörg Kindermann. Text categorization with support vector machines. how to represent texts in input space ? *Machine Learning*, 46 :423–444, 2002.

[lLS+00] Raj D. lyer, David D. Lewis, Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting for document routing. In *Proceedings of CIKM 2000*, pages 70–77, 2000.

[LMP97] Ludovic Lebart, Alain Morineau, and Marie Piron. *Statistique exploratoire multidimensionnelle, 2nd éd.* Dunod, 1997.

[LSCP96] David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 298–306, Zürich, Suisse, 1996. ACM Press.

[LW94] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2) :212–261, 1994.

[McC96] Andrew K. McCallum. Bow : A toolkit for statistical language modeling, text retrieval, classification and clustering. `http://www.cs.cmu.edu/~mccallum/bow`, 1996.

[Mil58] G. A. Miller. Tests of a statistical explanation of the rank-frequency relation for words in written english. *American Journal of Psychology*, 71 :209–218, 1958.

[Mit97] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[Mit98] Tom M. Mitchell. Conditions for the equivalence of hierarchical and flat bayesian classifiers. Technical report, Center for Automated Learning and Discovery, Carnegie- Mellon University, 1998.

[Mla98a] Dunja Mladenic. Feature subset selection in text-learning. In *European Conference on Machine Learning (ECML)*, pages 95–100, 1998.

[Mla98b] Dunja Mladenic. Turning Yahoo to automatic web-page classifier. In *European Conference on Artificial Intelligence*, pages 473–474, 1998.

[MLW92] Brij Masand, Gordon Linoff, and David Waltz. Classifying news stories using memory based reasoning. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59–65, 1992.

[MN98] Andrew McCallum and Kamal Nigam. A comparison of event models for Naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[MN99] Andrew McCallum and Kamal Nigam. Text classification by bootstrapping with keywords, EM and shrinkage. In *ACL Workshop for Unsupervised Learning in Natural Language Processing*, 1999.

[MNRS00] Andrew K. McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2) :127–163, 2000.

[Moc99] Kenrick Mock. Dynamic email organization via relevance categories. In *International Conference on Tools with Artificial Intelligence (ICTAI'99)*, pages 399–405, 1999.

[Mos03] Alessandro Moschitti. A study on optimal parameter tuning for Rocchio text classifier. In *Proceedings of the 25th European Conference on Information Retrieval Research (ECIR)*, Pisa, Italy, 2003.

[MRMN98] Andrew McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 359–367. Morgan Kaufmann Publishers, 1998.

[Mul64] C. Muller. *Essai de statistique lexicale : l'Illusion Comique de P. Corneille.* Klincksieck, 1964.

[NDa] NIST and DARPA. Text REtrieval Conference. `http://trec.nist.gov`.

[NDb] NIST and DARPA. Topic Detection and Tracking. `http://www.nist.gov/speech/tests/tdt/index.htm`.

[NJ01] Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers : A comparison of logistic regression and naive bayes. *Neural Information Processing Systems*, 2001.

[NLM99] Kamal Nigam, John Lafferty, and Andrew McCallum. Using Maximum Entropy for text classification. In *Proceeding of the IJCAI Workshop on Information Filtering*, pages 61–67, Stockholm, Sweden, 1999.

[NMTM00] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2-3) :103–134, 2000.

[Oar94] Douglas W. Oard. Neural Networks in Information Filtering and Retrieval, 1994.

[OK02] Constantin Orăsan and Ramesh Krishnamurthy. A corpus-based investigation of junk emails. In *Proceedings of The Third International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas de Gran Canaria, Spain, 2002.

[OLC+01] J-H Oh, K-S Lee, D-S Chang, C. Won Seo, and K-S Choi. TREC-10 experiments at KAIST : Batch filtering and Question Answering. In *Proceedings of The Tenth Text REtrieval Conference (TREC-10)*, pages 347–354, 2001.

[Pap01] Kishore Papineni. Why Inverse Document Frequency ? In *Proceedings of the NAACL 2001*, pages 25–32, USA, 2001.

[PLV02] Bo Pang, Lillian Lee, and Shivakumar Vaithyannathan. Thumbs up ? sentiment classification using Machine Learning techniques. In *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2002*, 2002.

[Por80] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3) :130–137, 1980.

[RdB01] Liva Ralaivola and Florence d'Alché Buc. Incremental Support Vector Machine learning : A local approach. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2001.

[Ren00] Jason D. M. Rennie. ifile : An application of machine learning to mail filtering. In *Proceedings of the KDD-2000 Workshop on Text Mining*, 2000.

[Ril95] Ellen Riloff. Little words can make a big difference for text classification. In *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 130–136, Seattle, US, 1995. ACM Press.

[RK98] Hein Ragas and Cornelis H. Koster. Four text classification algorithms compared on a Dutch corpus. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 369–370. ACM Press, 1998.

[Roc71] Joseph John Rocchio. *The SMART Retrieval System : Experiments in Automatic Document Processing*, chapter 14, Relevance Feedback in Information Retrieval, pages 313–323. Gerard Salton (editor), Prentice-Hall Inc., New Jersey, 1971.

[RS02] Miguel E. Ruiz and Padmini Srinivasan. Hierarchical text categorization using neural networks. *Information Retrieval*, 5(1) :87–118, 2002.

[Sal71] Gerard Salton, editor. *The SMART Retrieval System : Experiments in Automatic Document Processing*. Prentice-Hall Inc., New Jersey, 1971.

[Sal93] Marcos Salganicoff. Explicit forgetting algorithms for memory based learning. Technical Report MS-CIS-93-80, University of Pennsylvania, Dept of Computer and Information Science, 1993.

[Sal97] Steven Salzberg. On comparing classifiers : Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3) :317–328, 1997.

[SBM96] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the 19th ACM SIGIR Research and Development in Information Retrieval*, pages 21–29, 1996.

[SC01] Carl Sable and Ken Church. Using bins to empirically estimate term weights for text categorization. In *Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing*, pages 58–66, Pittsburgh, US, 2001. ACL.

[SDHH98] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization : Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[Seb02] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1) :1–47, 2002.

[SFBL97] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin : a new explanation for the effectiveness of voting methods. In *Proceedings of the 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann, 1997.

[SH99] Carl L. Sable and Vasileios Hatzivassiloglou. Text-based approaches for the categorization of images. In *Proceedings of ECDL-99, 3rd European Conference on Research and Advanced Technology for Digital Libraries*, pages 19–38. Springer Verlag, 1999.

[SHP95] Hinrich Schütze, David A. Hull, and Jan O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th ACM/SIGIR Conference*, pages 229–237, 1995.

[SK99a] Richard Segal and Jeffrey O. Kephart. MailCat : An intelligent assistant for organizing e-mail. In *International Conference on Autonomous Agents*, pages 276–282, 1999.

[SK99b] Richard B. Segal and Jeffrey O. Kephart. Incremental learning in swiftfile. In *Proceedings of the Seventh International Conference on Machine Learning (ICML'99)*, 1999.

[SK00] Shrikanth Shankar and Georges Karypis. Weight adjustment schemes for a centroid based classifier. Technical report, Department of Computer Science, University of Minnesota, Minneapolis, 2000.

[SKK00] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[SM83] Gerard Salton and Michael McGill. *Introduction to modern information retrieval.* McGraw-Hill, New York, 1983.

[SMB97] Amit Singhal, Mandar Mitra, and Christopher Buckley. Learning routing queries in a query zone. In *Proceedings of SIGIR-97, 20th ACM*

International Conference on Research and Development in Information Retrieval, pages 25–32, Philadelphia, US, 1997.

[Spi00] Martijn Spitters. Comparing feature sets for learning text categorization. In *Proceedings of RIAO'2000*, pages 1124–1135, Paris, 2000.

[SS90] G. W. Stewart and Ji-Gang Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

[SS00] Robert E. Schapire and Yoram Singer. BoosTexter : A boosting system for text classification. *Machine Learning*, 39(2/3) :135–168, 2000.

[SSS98] Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting and Rocchio applied to text filtering. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 215–223, Melbourne, AU, 1998. ACM Press.

[ST01] N. Slonim and N. Tishby. The power of word clusters for text classification. In *Proceedings of the 23rd European Colloquium on Information Retrieval Research (ECIR)*, 2001.

[Str00] Mathieu Stricker. *Réseaux de neurones pour le traitement automatique du langage : conception et réalisation de filtres d'information*. PhD thesis, ESPCI, 2000.

[SWY75] Gerard Salton, A. Wong, and C.S. Yang. A Vector Space Model for Information Retrieval. *Communications of the ACM*, 18(11) :613–620, November 1975.

[TCPH01] Kristina Toutanova, Francine Chen, Kris Popat, and Thomas Hofmann. Text classification in a hierarchical mixture model for small training sets. In *Proceedings of the Tenth International ACM Conference on Information and Knowledge Management (CIKM)*, pages 105–113, 2001.

[TK00] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. In *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 999–1006. Morgan Kaufmann Publishers, 2000.

[Tur02] Peter Turney. Thumbs up or thumbs down ? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, Philadelphia, 2002.

[Utg89] Paul Utgoff. Incremental induction of decision trees. *Machine Learning*, 4(2) :161–186, 1989.

[Vap95] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[VGV03] Romain Vinot, Natalia Grabar, and Mathieu Valette. Application d'algorithmes de classification automatique pour la détection des contenus racistes sur l'Internet. In *Actes de la 10ème conférence sur le Traitement Automatique du Langage Naturel (TALN)*, pages 275–284, 2003.

[VR03] Ricardo Vilalta and Irina Rish. Decomposition of classes via clustering to explain and improve Naïve Bayes. In Springer, editor, *Proceedings of the 14th Eurpoean Conference on Machine Learning (ECML)*, pages 444–455, Cavtat-Dubrovnik, 2003.

[VY01] Romain Vinot and François Yvon. Semi-automatic response in a Mail Center. In *ASMDA 2001, 10th International Symposium on Applied Stochastic Models and Data Analysis*, pages 992–997. Université de Technologie de Compiègne, 2001.

[VY02] Romain Vinot and François Yvon. Quand simplicité rime avec efficacité : Analyse d'un catégoriseur de textes. In *Colloque International sur la Fouille de Texte (CIFT'02)*, pages 17–26, Tunisie, 2002.

[WAD+99] Sholom M. Weiss, Chidanand Apt, Fred J. Damerau, David E. Johnson, Frank J. Oles, Thilo Goetz, and Thomas Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4) :63–69, 1999.

[Wei99] Yair Weiss. Segmentation using eigenvectors : A unifying view. In *International Conference on Computer Vision (ICCV)*, pages 975–982, 1999.

[WF00] Wai-Chiu Wong and Ada Fu. Incremental document clustering for web page classification. In *IEEE 2000 International Conference on Information Society in the 21st century : emerging technologies and new challenges (IS2000)*, 2000.

[Wil88] P. Willet. Recent trends in hierarchical document clustering : a critical review. *Information Processing and Management*, 24 :577–597, 1988.

[WK96] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23 :69–101, 1996.

[WPW95] Erik D. Wiener, Jan O. Pedersen, and Andreas S. Weigend. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 317–332, Las Vegas, US, 1995.

[WW99] J. Weston and C. Watkins. Support vector machines for multiclass pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks (ESAN)*, pages 219–224, 1999.

[WZL99] Ke Wang, Senqiang Zhou, and Shiang Chen Liew. Building hierarchical classifiers using class proximity. In *Proceedings of 25th International Conference on Very Large Data Bases (VLDb'99)*, pages 363–374, 1999.

[Yan94] Yiming Yang. Expert Network : Effective and efficient learning from human decisions in text categorization and retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 13–22, 1994.

[Yan99] Yiming Yang. An evaluation of statistical approach to text categorization. *Journal of Information Retrieval*, 1(1/2) :67–88, 1999.

[YC94] Yiming Yang and Christopher Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12(3) :252–277, 1994.

[YP97] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. 14th International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann, 1997.

[YPC98] Yiming Yang, Tom Pierce, and Jaime Carbonell. A study on retrospective and on-line event detection. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 28–36, 1998.

[YZK03] Yiming Yang, Jian Zhang, and Bryan Kisiel. A scalability analysis of classifiers in text categorization. In *Proceedings of the 26th Conference on Research and Development in Information Retrieval (SIGIR)*, pages 96–103, 2003.

[ZH01] Sarah Zelikovitz and Haym Hirsh. Improving text classification with LSI using background knowledge. In *IJCAI01 Workshop Notes on Text Learning : Beyond Supervision*, 2001.

[ZO01] Tong Zhang and Frank J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1) :5–31, 2001.

[ZY03] Jian Zhang and Yiming Yang. Robustness of regularized linear classification methods in text categorization. In *26th Aunnual International ACM SIGIR Conference (SIGIR 2003)*, pages 190–197, Canada, 2003.