Chapitre IV. Classification de texte et apprentissage automatique: État de L'art

1. Introduction

Le domaine de 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 vue de la quantité phénoménale de textes accessibles de nos jours (par le biais du réseau Internet en particulier), il devient urgent de trouver des solutions opérationnelles pour automatiser le plus grand nombre possible de taches 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 [Vinot, 2004].

Le but n'est pas ici de fournir un système de traitement automatique mais plutôt un ensemble d'outils que les experts puissent l'utiliser pour découvrir des concordances ou des tendances générales difficilement décelables à la main. La deuxième approche consiste à proposer des systèmes de type «boite noire» qui traite les données textuelles de façon automatique sans intervention humaine. Il est bien nécessaire de clarifier les termes qui seront employés dans les sections qui suivent. La catégorisation correspond donc à la classification supervisée pour l'apprentissage automatique et à la discrimination pour la statistique alors que la recherche d'information emploie des termes plus proches à l'application concernée: filtrage ou routage.

La deuxième section de ce chapitre introduit le problème de classification automatique de texte. Dans la troisième section, nous présenterons les différentes stratégies de représentation des documents traités par un classificateur. Les techniques de sélection et d'extraction d'attributs seront aussi exposées dans cette section. Dans la quatrième section, nous détaillerons le fonctionnement des classificateurs les plus connus dans ce domaine. La cinquième section abordera les critères d'évaluation des classificateurs.

2. La catégorisation des textes

2.1 Définition de la catégorisation des textes

La Catégorisation de Textes est une tache générique qui englobe le filtrage. Elle consiste à assigner une ou plusieurs catégories parmi une liste prédéfinie à un document donné. Ces catégories correspondent dans la plupart des cas aux sujets principaux 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).

2.2 Description de la catégorisation de textes

Le processus de catégorisation intègre la construction d'un modèle de prédiction qui, en entrée, reçoit un texte et, en sortie, lui associe une ou plusieurs étiquettes. Pour identifier la catégorie ou la classe à laquelle un texte est associé, un ensemble d'étapes est habituellement suivies. Ces étapes concernent principalement la manière dont un texte est représenté, le choix

de l'algorithme d'apprentissage à utiliser et comment évaluer les résultats obtenus pour garantir une bonne généralisation du modèle appris [Jalam, 2003].

Soit C=$\{c_1, c_2, c_3, \dots c_n\}$ un ensemble de catégories ou de classes et D= $\{d_1, d_2, d_3, \dots d_N\}$ un ensemble de documents. Á chaque document di est associée une classe ci à laquelle il appartient. L'objectif est d'apprendre une fonction $f: D \rightarrow C$ tel que pour tout couple (di, ci), f (di) = ci. La fonction f est appelée classificateur.

Cette fonction peut être construite manuellement ou automatiquement à partir d'un ensemble de couples document-catégorie. Le processus de catégorisation, intégrant la phase de classement de nouveaux textes, est résumé dans la figure 4.1. Il comporte deux phases que l'on peut distinguer comme suit :

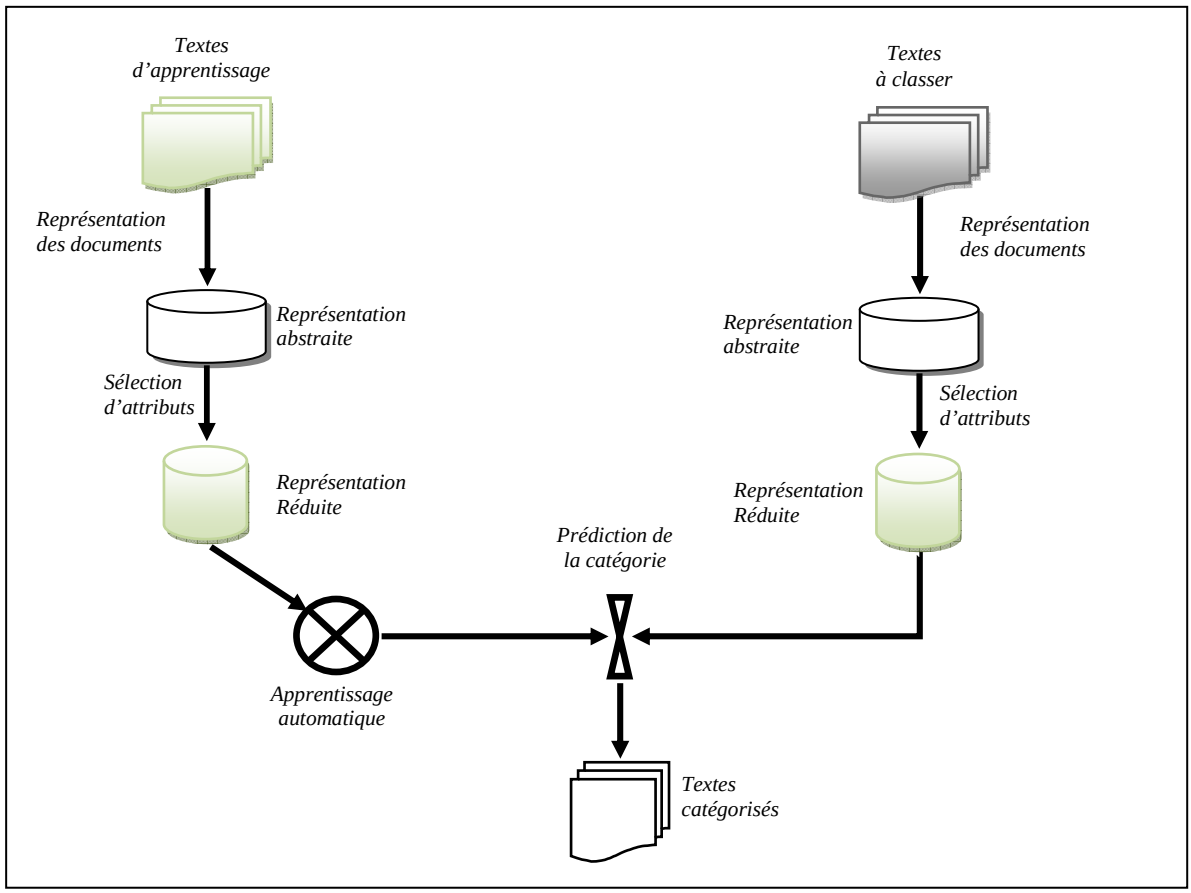

Fig. 4.1. Processus de catégorisation de texte [Jalam 2003]

2.2.1. La phase d'apprentissage «Learning»

L'apprentissage automatique se divise principalement en deux façons d'apprendre: l'apprentissage supervisé et l'apprentissage non supervisé. Dans le cas manuel comme le système *Construe* [Hayes et Weinstein, 1990], 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 x est présent en même temps que le mot y alors la classe est z».

La détermination de cette fonction f est appelée phase d'apprentissage. L'inconvénient de cette approche est que l'édition des règles de décision peut s'avérer très long. Et surtout, si des catégories s'ajoutent ou si on désire utiliser le classificateur dans un autre domaine, on doit répéter l'exercice [Réhal, 2005]. Selon [Vinot, 2004], dans le cas automatique c'est un algorithme

qui analyse le corpus. La fonction calculée f n'est pas toujours intelligible «claire» par des humains car elle peut faire intervenir un grand nombre de valeurs numériques qu'un humain ne peut pas appréhender. f est ensuite utilisée pour attribuer une catégorie à tout nouveau document, dans la phase de classification.

2.2.2. La phase de classification « test»

Après la phase d'entraînement, le classificateur peut procéder lui-même au classement de nouveaux textes. Plus précisément, on construit un classificateur binaire pour chaque catégorie, qui va déterminer si, oui ou non un document en fait partie. Notons que les descripteurs les plus pertinents sont extraits lors de la première phase par analyse des textes du corpus d'apprentissage. Dans la seconde phase, celle du classement d'un nouveau texte, nous cherchons simplement la fréquence de ces descripteurs dans ce texte à classer.

Dans la suite de ce chapitre, nous présentons en détail les étapes de sélection d'attributs et les différents algorithmes de catégorisation de texte.

3. Représentation des documents

D'après [Salton, 1983] les algorithmes d'apprentissage ne sont pas capables de traiter directement les textes ni, plus généralement, les données non structurées comme les images, le son et les séquences vidéos. C'est pourquoi une étape préliminaire dite de représentation est nécessaire. Cette étape consiste généralement en la représentation de chaque document par un vecteur, dont les composantes sont par exemple les mots contenus dans le texte, afin de le rendre exploitable par les algorithmes d'apprentissage.

Selon [Vinot, 2004] la quasi-totalité des systèmes de catégorisation 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 : chaîne de caractères, mots, racines des mots, groupe 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 «*bag-of-words*».

Un document d_i est alors représenté par un vecteur d'attributs $d_i = (d_{i1},...d_{ik},..., d_{iv})$, ou 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. La représentation des termes englobe trois grandes fonctions à savoir: le prétraitement des documents, la réduction de la taille du vocabulaire et la pondération des termes.

3.1 Prétraitement des documents (voir aussi Chapitre 1, Section 3)

Dans tous les cas (utilisation de mots simple ou de groupes de mots), il est préférable d'effectuer quelque prétraitement afin de filtrer les mots non informatifs et de regrouper les mots de la même famille.

La première opération «élimination des mots vides» consiste à supprimer les mots faisant partie d'une liste prédéfinie : liste des mots vides (*Stopwords*). Ce sont des mots génériques non porteurs de sens tels que les déterminants, les articles (le, la, les) et les auxiliaires qui sont a priori inutiles pour discriminer les différentes catégories. Ils peuvent donc être supprimés sans perte d'information utile.

La deuxième opération «lemmatisation/Racinisation» 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 :

- De prendre en compte les variations flexionnelle (singulier/pluriel, conjugaison,...) ou dérivationnelles (substantifs, verbes ou adjectives) en regroupant sous le même terme tous les mots de la même famille, par exemple «étude», «étudiant» et «étudier» pourraient être regroupés, et un seul terme serait ajouté à l'espace vectoriel plutôt que trois [Réhal, 2005].
- De baisser le nombre total de termes et donc les temps de calcul.
- Dans le cas de la Racinisation, on utilise des heuristiques simples à partir de règles de remplacement des chaînes de caractères pour supprimer les affixes (suffixes et préfixes) des mots et extraire la racine [Vinot, 2004].

3.2 Réduction de la taille du vocabulaire

Malgré ces premières transformations, la taille du vocabulaire peut encore s'avérer très importante. Il est souvent préférable de la diminuer davantage avant d'appliquer les techniques de classification les plus complexe.

3.2.1. Définition d'attribut «terme»

On peut le considérer comme étant une suite de caractères appartenant à un dictionnaire, ou bien, de façon plus pratique, comme étant une séquence de caractères non-délimiteurs encadrés par des caractères délimiteurs (caractères de ponctuation) [Jalam, 2003].

3.2.2. Sélection d'attributs «*Feature Selection*» [Massih, 2007]

Les techniques de sélection visent à retenir les termes les plus informatifs en s'appuyant sur les statistiques des corpus. En classification deux questions fondamentales se posent avant l'application de ces techniques:

- De combien faut-il réduire la taille des attributs sans perdre l'information?
- Jusqu'à quelle limite la sélection des attributs peut-elle augmenter la performance d'un classificateur ?

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 [Vinot, 2004].

En générale on utilise différents critères de pertinence de termes pour éliminer ceux qui sont les moins informatifs. Parmi ces techniques, on trouve [Vinot, 2004] [Massih, 2007]:

- La fréquence du document ou «*Document Frequency*»: elle est très simple à calculer puisqu'elle correspond simplement au pourcentage de document dans lesquels le terme apparaît (cette technique tend à supprimer les termes qui apparaissent rarement).
- Le Gain d'information: mesure le nombre de bits d'information obtenue par la classification en connaissant la présence ou l'absence d'un terme dans un document.
- L'information mutuelle: utilisée généralement pour modéliser la corrélation (lien) entre deux variables aléatoires, ici le terme et la catégorie.
- Le test de Chi-deux (χ^2): mesure le degré d'indépendance entre un terme et une classe.

Les formules mathématiques de ces techniques sont illustrées dans le tableau suivant :

Technique de sélection d'attributs	Formule mathématique
Fréquence-document (DF)	$P(tk)$
Gain d'information (GI)	$P(t_k, c_j)\log\frac{P(t_k, c_j)}{P(t_k)P(c_j)} + P(t_k, c_j)\log\frac{P(\bar{t_k}, c_j)}{P(\bar{t_k})P(c_j)}$
Information mutuelle (IM)	$\log\frac{P(t_k, c_j)}{P(t_k)P(c_j)}$
Chi-deux (X^2)	$\frac{\lvert N\rvert\left[P(t_k, c_j)P(\bar{t_k}, \bar{c_j}) - P(t_k, \bar{c_j})P(\bar{t_k}, c_j)\right]^2}{P(t_k)P(\bar{t_k})P(c_j)P(\bar{c_j})}$

Tableau 4.1. Différentes mesures de la quantité d'informations contenues dans le terme t_k pour la classec_j. [Jalam 2003]

$P(t_k, c_j)$ est la probabilité pour un document donné que le terme t_k soit présent et que le document soit de la classe c_j. Elle estimée par le rapport du nombre de documents de c_j dans lequel t_k apparaît sur le nombre total de documents. $\bar{A}$ indique l'absence de l'événement A. $P(\bar{t}_k, \bar{c}_j)$ est donc la probabilité que le terme t_k soit absent et que le document ne soit pas de la classe c_j (ainsi de suite pour les autres probabilités). Le Tableau 4.2 ci-dessous représente une comparaison entre les différentes techniques de sélection des termes (attributs) [Massih, 2007].

Technique	DF	IM	X^2	GI
utilise l'absence des termes	*Non*	*Non*	*Oui*	*Oui*
utilise l'information de classe	*Non*	*Oui*	*Oui*	*Oui*
favorise les termes courants	*Oui*	*Non*	*Oui*	*Oui*

Tableau 4.2. tableau comparatif des techniques de sélection d'attributs.

3.2.3. Extraction d'attributs «*Feature Extraction*»

Une alternative à la sélection d'attributs pour la réduction de l'espace vectoriel est l'extraction d'attributs. Ce processus consiste à créer à partir des attributs originaux un ensemble d'attributs synthétiques qui maximisent l'efficacité de la classification. Le regroupement de termes «*Term Clustering*» fait partie de ces techniques et a pour but de grouper les mots qui ont une sémantique commune. Les groupes «clusters» ainsi créés deviennent les attributs d'un nouvel espace vectoriel. Pour déterminer ces groupes, il faut utiliser un «clustering» avec une mesure de similarité entre mots à partir des profils d'occurrence des termes dans les différentes classes. Sans entrer dans les détails, mentionnons seulement que l'idée de départ est la suivante : si deux mots différents apparaissent dans les mêmes classes et dans les mêmes proportions, les réunir en un seul attribut qui affiche la distribution moyenne ne peut pas affecter négativement la performance de la classification [Réhel, 2005].

3.3 Pondération d'attributs «calcul des poids»

Une fois la liste des attributs est 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. La pondération s'appuie sur deux notions : La Fréquence du Terme «*Terme Frequency*» ou *TF* qui prend en compte le nombre d'occurrences du terme dans le document et l'Inverse de la fréquence en document «*Inverse Document Frequency*» ou *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. Ainsi la formule de TFIDF sera comme suit :

$$TFIDF(t,d) = TF(t,d) * IDF(t,d) = TF(t,d) * \log_2(\frac{N}{DF} + 1) \quad (4.1)$$

G Où : *t* est un terme, *d* est un document, *TF(t, d)* est le nombre d'apparitions de *t* dans *d*, *DF* est le nombre de documents du corpus qui contiennent *t* et *N* est le nombre total de documents dans le corpus.

4. Algorithmes de classification automatique des textes

Les algorithmes d'apprentissage supervisés tentent d'ajuster un modèle qui explique le lien entre des documents d'entrée et les classes de sortie. En catégorisation de textes, on fournit à la machine des exemples sous la forme (Document, Classe). Cette méthode de raisonnement est appelée inductive car on induit de la connaissance (le modèle) à partir des données d'entrée (l'échantillon de Documents) et des sorties (leurs Classes). Grâce à ce modèle, on peut alors estimer les classes de nouveaux documents : le modèle est utilisé pour « prédire ». Le modèle est bon s'il permet de bien prédire.

Différents types de classificateurs ont été mis au point, toujours dans le but d'atteindre un degré maximal de précision et d'efficacité, chacun ayant ses avantages et inconvénients. Ils partagent toutefois des caractéristiques communes. Le nombre important des algorithmes de catégorisation de textes rend impossible une présentation exhaustive de ces derniers. Dans cette section nous nous contentons de présenter les méthodes les plus utilisées dans la littérature en insistant sur les caractéristiques, les avantages et les limites de chaque méthode.

4.1 Classificateur Rocchio

[Rocchio, 1971] 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. La technique *Rocchio* a été adaptée à la catégorisation de textes. L'algorithme calcule un profil prototypique P_j (c'est-à-dire un vecteur de l'espace de représentation des documents) pour chaque classe c_j qui correspond au barycentre de document positif et négatif (avec un coefficient négatif pour les documents qui n'appartiennent pas a c_j):

$$P_{jk} = max\left\{\frac{t}{|c_j|}\sum_{d_i \in c_j} d_{ik} - \frac{1-t}{|\overline{c_j}|}\sum_{d_i \notin c_j} d_{ik}\right. \quad (4.2)$$

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

$$f(d_i) = arg\max_{c_j \in c}\left(\sin\left(d_i, c_j\right)\right) = arg\max_{c_j \in c}\left[\sum_{tk} d_{ik} p_{jk}\right] \quad (4.3)$$

4.1.1. 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. Les profils sont des vecteurs

comme les documents. Dans le cas ou t = 1, le profil P_j est alors le barycentre des exemples de la classe. Lorsque $t \neq 1$ les moyennes des exemples positifs et négatifs sont calculés 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.

L'algorithme de base *Rocchio* n'a pas été amélioré de façon probante et il est désormais très en deçà des classificateurs les plus performants. Mais grâce à son efficacité en termes de vitesse d'exécution, il est actuellement le classificateur le plus utilisé dans les logiciels commerciaux de catégorisation de textes.

4.2 Classificateur bayesien naïf

Le classificateur bayesien naïf est le représentant le plus populaire des classificateurs 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. Dans cet algorithme dont le modèle d'apprentissage est très général, la classification d'un exemple (un document) s'obtient par estimation de $P(c_j/d_i)$, la probabilité connaissant l'exemple *di* que celui-ci fasse partie de la classe *cj*. 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 (multi-classe). La formule de bayes permet ″d'inverser″ la probabilité conditionnelle :

$$P(c_j|d_i) = \frac{P(c_j) * P(d_i|c_j)}{P(d_i)} \qquad (4.4)$$

Le calcule de $P(d_i/c_j)$ dépend du modèle de génération des exemples. Dans le cas du texte, le modèle le plus populaire est le modèle multivarié de Bernoulli [Lewis et al., 1998].

4.2.1. Le modèle multivarié de Bernoulli

Dans le modèle le plus simple, un document est un vecteur binaire de la taille du vocabulaire. Chaque terme du vocabulaire peut être présent ou absent avec une certaine probabilité. Seule la présence /absence des termes est utilisée, leur nombre d'occurrence dans le document n'a pas d'incidence. Pour rendre les formules exploitables, on fait de plus l'hypothèse d'indépendance des termes (hypothèse naïve de Bayes). $P(d_i/c_j)$ se simplifie alors en un produit de probabilités d'apparitions de chaque terme :

$$P(d_i|c_j) = \prod_{t_{ik} \in d_i} P(t_k|d_i) \qquad (4.5)$$

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 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 [Vinot, 2004]. C'est pourquoi le lissage effectué pendant l'estimation est indispensable. On obtient alors dans le cas du lissage de Laplace :

$$P(t_k|c_j) = \frac{1 + \sum_{d_i \in c_j} 1_{ik}}{2 + |c_j|} \qquad (4.6)$$

Avec : 1ik = 1 si tk appartient di 0 sinon ;

Le fonctionnement du modèle bayesien naïf 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 bayesien naïf). Tous ces coefficients sont ensuite regroupés pour former une valeur de pertinence (un degré de similarité pour Rocchio, une probabilité pour bayesien naïf). Il manque néanmoins au bayesien naïf une meilleure prise en compte de la taille des documents et de l'importance logarithmique des fréquences de termes. C'est pourquoi, le modèle bayesien naïf obtient en général des performances légèrement inferieures à Rocchio [Yang, 1999].

4.3 Classificateur K plus proches voisins

L'algorithme des *K* plus proches voisins est un des plus vieux algorithmes de la reconnaissance des formes [Cover & Hart, 1967]. Il a été employé avec succès dans de nombreux domaines et a engendré toute une famille de classificateurs connus sous le nom de classificateurs paresseux (*Lazy Learners*). 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 a la phase de classification (d'où le terme de paresseux). Dans les *K-PPV*, pour chaque nouvel exemple d à catégoriser, le classificateur 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 leurs similarités) de ces *K* documents, donnant lien aux formules suivantes :

$$f(d) = arg\max_{c_j \in C}\left[\sum_{i=1}^{k} C\left(d_i, c_j\right)\right] \quad (4.7)$$

$$f(d) = arg\max_{c_j \in C}\left[\sum_{i=1}^{k} sim(d, d_i) * C\left(d_i, c_j\right)\right] \quad (4.8)$$

Avec $C(d_i, c_j) = 1$ si d_i est de classe c_j, 0 sinon [Vinot, 2004]. L'algorithme générique de classification d'un nouvel exemple par la méthode KNN est le suivant:

Paramètre : le nombre *k* de voisins
Donnée : un échantillon de m enregistrements classés (*x, c(x)*)
Entrée : un enregistrement y
1. déterminer les *k* plus proches enregistrements de *y*
2. combiner les classes de ces *k* exemples en une classe *c*

Sortie : la classe de *y* est *c(y)=c*.

Algorithme 4.1. Algorithme de classification par *K-PPV* [Gilleron & Tommasi, 2000].

Dans le *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. 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 de classification. Ce problème est d'autant plus important dans le cas du texte où les documents sont dans un espace de très grande dimension (le calcule de la similarité prend dans ce cas un temps non négligeable) [Vinot, 2004].

4.4 Machines à support vectoriel

Les machines à support vectoriel («*Support Vector Machines*» ou *SVM*) [Vapnik, 1995] forment une classe d'algorithmes d'apprentissage qui peuvent s'appliquer à tout problème qui implique un phénomène *f* et qui, à partir d'un jeu d'entrées *x*, produit une sortie *y = f(x)*, et où le but est de retrouver *f* à partir de l'observation d'un certain nombre de couples entrée/sortie. Le

problème revient à trouver une frontière de décision qui sépare l'espace en deux régions, à trouver l'hyperplan qui classifie correctement les données et qui se trouve le plus loin possible de tous les exemples. On dit qu'on veut maximiser la marge, la marge étant la distance du point le plus proche de l'hyperplan.

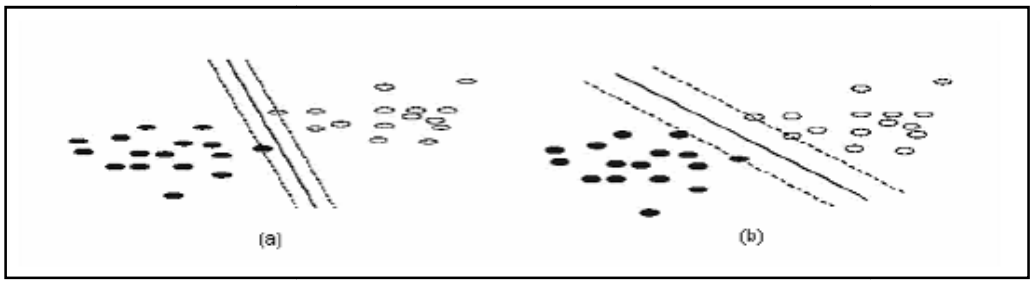

Fig. 4.2. Maximisation de la marge avec les SVM, (a) une frontière possible séparant les données, (b) la frontière entraînant une marge maximale.

Dans la figure 4.2, la partie de droite représente la frontière recherchée entre les données. Il s'agit de la ligne en trait continu. La marge, soit la distance entre cette ligne et les lignes en trait pointillé, y est maximale. La partie de gauche présente une situation où la marge serait plus petite. Les points situés sur les lignes en trait pointillé sont ceux qu'on appelle les vecteurs de support. Des techniques de programmation quadratique permettent de résoudre ce problème, un algorithme efficace étant présenté entre autres dans [Platt, 1999]. Une propriété intéressante des *SVM* est que la surface de décision est déterminée uniquement par les points qui en sont les plus près, les vecteurs de support. En présence de ces seuls exemples d'entraînement, la même fonction serait apprise. C'est une différence par rapport à des algorithmes comme *KNN* avec lesquels tous les exemples d'entraînement sont utilisés lors de l'apprentissage [Yang, 1999].

Même s'ils cherchent l'hyperplan séparant l'espace vectoriel en deux, l'avantage des *SVM* est qu'ils s'adaptent facilement aux problèmes non linéairement séparables. Avant de procéder à l'apprentissage de la meilleure séparation linéaire, on transforme les vecteurs d'entrée en vecteurs de caractéristiques de dimension plus élevée. De cette façon, un séparateur linéaire trouvé par un *SVM* dans ce nouvel espace vectoriel devient un séparateur non linéaire dans l'espace original. Cette transformation des vecteurs se fait à l'aide de noyaux "*kernels*". Dans le cas de la classification de textes, les entrées sont des documents et les sorties sont des catégories. En considérant un classificateur binaire, on voudra lui faire apprendre l'hyperplan qui sépare les documents appartenant à la catégorie et ceux qui n'en font pas partie. Selon [Joachim, 1998], les *SVM* conviennent bien pour la classification de textes parce que, premièrement, une dimension élevée ne les affecte pas puisqu'ils se protègent contre le sur-apprentissage. Dans le même sens, il affirme que peu d'attributs sont totalement inutiles à la tâche de classification et que les *SVM* permettent d'éviter une sélection agressive qui aurait comme résultat une perte d'information. On peut se permettre de conserver plus d'attributs. Également, une caractéristique des documents textuels est que lorsqu'ils sont représentés par des vecteurs, une majorité des entrées sont nulles. Or, les *SVM* conviennent bien à des vecteurs dits clairsemés. Un autre aspect positif des *SVM* est qu'aucun ajustement de paramètres manuel n'est requis, car ils ont l'habilité de trouver automatiquement des paramètres adéquats [Réhel, 2005].

L'apprentissage des SVM pose toutefois le problème du réglage d'au moins deux paramètres : le paramètre de pénalité C et le paramètre du noyau (dans le cas d'un noyau RBF). Le réglage de ces paramètres est souvent appelé (sélection de modèle) et est généralement réalisé par essai/erreur sur une base de test ou par validation croisée.

L'autre inconvénient des SVM concerne leur comportement en haute dimension. Si théoriquement les SVM supporte bien les hautes dimensions, dans la pratique on voit leurs performances chuter. C'est la raison pour laquelle des méthodes de sélection de variables sont souvent mises en œuvre [Huang et al., 04].

5. Evaluation des performances des classificateurs

Dans le domaine de la recherche d'information et de l'apprentissage automatique, la validation des méthodes est généralement empirique. Il existe plusieurs grandes compagnes d'évaluation internationales, chacune se focalisant sur une tache particulière dont le but est de comparer objectivement différents systèmes sur les mêmes données et avec les mêmes mesures de performance. Malgré ces compagnes, il reste très difficile d'affirmer la supériorité d'un algorithme sur un autre car les résultats sont très dépendants des corpus utilisées pour la tache à évaluer, des mesures de performance et même des implémentation [Salton , 1983].

5.1 Critères d'évaluation des classificateurs

Presque toujours, on divise le corpus de textes déjà classées et disponible en deux ensembles, l'ensemble d'entrainement sur lequel le classificateur fait son apprentissage et l'ensemble de test sur lequel on peut évaluer sa performance. Ils existent de nombreuses mesures pour calculer la performance d'un classificateur. Elles dépendent essentiellement du modèle de tache pour lequel le système de catégorisation de texte (CT) est utilisé. Pour mieux illustrer les différentes mesures qui vont suivre, on prend pour point de départ la table de contingence illustré par le tableau ci-dessous, distincte pour chaque catégorie.

	Document appartenant à la catégorie	Document n'appartenant pas à la catégorie
Document assignés à la catégorie par le classificateur	*a*	*b*
Document rejetés de la catégorie par le classificateur	*c*	*d*

Tableau 4.3. table de contingence à la base de l'évaluation des classificateurs

On définit à partir des statistiques de cette table les mesures suivantes :

Précision :

$$Précision = a / (a + b) \quad (4.9)$$

Soit le nombre d'assignations correctes sur le nombre total d'assignations.

Rappel :

$$Rappel = a/ (a + c) \quad (4.10)$$

Soit le nombre d'assignations correctes sur le nombre d'assignation qui auraient du être faites.

Exactitude :

$$Exactitude = (a + d) / (a + b + c + d) \quad (4.11)$$

Erreur :

$$Erreur = (b + c) / (a + b + c + d) \quad (4.12)$$

Les deux dernières mesures, bien que couramment utilisées en apprentissage automatique, sont jugées moins adaptées à la tache de classification de textes. La précision et le rappel sont les mesures les plus rencontrées dans la littérature. Lors de l'évaluation de la performance d'un classificateur, on ne peut tenir compte de la précision ou de rappel séparément. Effectivement,

on pourrait mettre en place un système qui rejetterait tous les textes: il obtiendrait une précision de 100%, mais un rappel de 0%. A l'inverse, un système qui accepterait tous les textes aurait un rappel de 100%, mais une précision de 0%. On voit donc qu'un meilleur classificateur est celui qui tente de faire le compromis idéal entre ces deux facteurs. Aussi, la mesure *F1* est beaucoup utilisée. Elle est définie ainsi :

$$\mathrm{F1} = \frac{2rp}{(r+p)} \tag{4.13}$$

Où : r représente le rappel et p représente est la précision, F1est une fonction qui est maximisée quand la précision et le rappel sont proches [Réhal, 2005].

6. Conclusion

Ce chapitre a permis de donner un état de l'art global sur la classification automatique de textes. Tout d'abord, il s'attarde sur la nature du problème à résoudre. Par la suite, il a été question de différentes stratégies de représentation des documents traités par un classificateur. Le choix judicieux d'un mode de représentation des données est nécessaire, comme pour toute application de l'apprentissage automatique. Enfin, des techniques de sélection et d'extraction d'attributs ont été exposées. Celles-ci visent à réduire la taille du vocabulaire à traiter pour que les algorithmes évoluent dans un espace vectoriel de dimension raisonnable. Dans la deuxième partie du chapitre, on a pu constater la variété de techniques d'apprentissage pouvant amener une application informatique à classer des textes avec autonomie. De façon succincte, on comprend maintenant mieux le fonctionnement des classificateurs Rocchio, bayesien, k- plus proches voisins et machines à support vectoriel. En plus, la troisième partie du chapitre a fait la lumière sur le processus d'évaluation des classificateurs. Les chapitres suivants seront consacrés à l'introduction des deux modèles de catégorisation de texte retenues dans nos travaux. Nous étudierons le comportement des algorithmes arbres de décision et réseaux de neurones artificiels tels qu'ils sont utilisés pour la catégorisation de texte.