

CHAPITRE 3

LES ALGORITHMES DE CATEGORISATIONS

3.1. Introduction :

En apprentissage automatique, différents types de classificateurs ont été mis au point, et cela dont le but d'atteindre un degré maximal de précision et d'efficacité, chacun ayant ses avantages et ses inconvénients. Mais, ils partagent toutefois des caractéristiques communes.

Parmi la panoplie de classificateurs existants, on peut faire des regroupements et distinguer des grandes familles.

Dans les pages qui suivent, nous allons exposer quelques algorithmes en détail, l'algorithme de Rocchio et l'algorithme des k-voisins les plus proches. Puis, les machines à support vectoriel et réseaux de neurones et en fin le classifieur bayésien naïf algorithme surpassé par d'autres mais il est souvent utilisé comme point de référence à cause de sa simplicité (j'utilise cet algorithme comme une méthode probabiliste dans l'approche statistique pour l'identification de langue de mes documents textuels).

3.2. Différentes méthodes d'apprentissage :

3.2.1. Méthode de Rocchio :

La méthode de Rocchio est un classifieur linéaire proposé dans (Rocchio, 1971) , un des plus vieux algorithmes de classification destiné à l'amélioration des systèmes de recherche documentaires.

L'avantage de ce type de classifieur est la simplicité et l'interprétabilité. Pour un expert, ce profil prototype est plus compréhensible qu'un réseau de neurones par exemple. L'apprentissage de ce type de classifieur est souvent précédé par une sélection et une réduction de termes.

Ce classifieur s'appuie sur une représentation vectorielle des documents : chaque document d_j est représenté par un vecteur d_j de R^n (n est le nombre de termes après sélection et réduction); chaque coordonnée t_{kj} se déduit du nombre d'occurrences $\#(t_k, d_j)$ du terme t_k dans d_j , par [04] :

$$TF.IDF(t_k, d_j) = \log \left(1 + \#(t_k, d_j) \right) * \log |Tr| / (\#Tr(t_k)) \quad (1)$$

Avec :

- $\#(t_k, d_j)$: Nombre d'apparition de terme t_k dans le document d_j .
- $|Tr|$: le nombre de documents du corpus d'apprentissage.
- $\#Tr(t_k)$: le nombre de documents dans lesquels apparaît au moins une fois le terme t_k .

Un terme t_k se voit donc attribuer un poids d'autant plus fort qu'il apparaît souvent dans le document et rarement dans le corpus complet. Chaque vecteur d_j est ensuite normalisé, par la normalisation en cosinus, afin de ne pas favoriser les documents les plus longs [04].

$$t_k = TF.IDF(t_k, d_j) / \sqrt{\sum_{j=1}^m (TF.IDF(t_k, d_j))^2} \quad (2)$$

Selon la méthode de Rocchio, Chaque catégorie est représentée par un profil «prototypique». Un profil de la classe c_i est une liste de termes pondérés, dont la présence et l'absence discriminent au mieux cette classe c_i [22].

Pour chaque catégorie c_i , les coordonnées t_{ki} du profil prototypique $c_i = (t_{1i}, \dots, t_{|T|i})$ sont calculé ainsi :

$$C_i = t_{k_i} = \left(\frac{t}{N_c} * d_c \right) * (d_i - (1 - t) / (N_c^- * d_c)) * d_i \quad (3)$$

Où :

N_c : Le nombre de documents dans C.

N_c^- : Le nombre de documents n'appartenant pas à C.

t : Un paramètre du modèle compris entre 0 et 1. Dans les situations où un document peut être attribué à une seule classe, t est souvent positionné à 1.

d_i : Poids des termes dans les documents de la classe i .

C_i : Profil prototypique de la classe i .

Les profils prototypes c_i correspondent donc aux barycentres des exemples (avec un coefficient positif pour les exemples de la classe, et négatif pour les autres). Le classement de nouveaux documents s'opère en calculant la distance euclidienne entre la représentation vectorielle du document et celle de chacune des classes ; le document est assigné à la classe la plus proche [22].

3.2.2. Algorithme des k-plus proches voisins :

L'algorithme des k-voisins les plus proches («*k-nearest neighbors*» ou **kNN**) est une méthode d'apprentissage à base d'instances.

La méthode ne nécessite pas de phase d'apprentissage; c'est l'échantillon d'apprentissage, associé à une fonction de distance et à une fonction de choix de la classe en fonction des classes des voisins les plus proches, qui constitue le modèle.

Lorsqu'un nouveau document à classer arrive, il est comparé aux documents d'entraînement à l'aide d'une mesure de similarité. Ses k plus proches voisins sont alors considérés : on observe leur catégorie et celle qui revient le plus parmi les voisins est assignée au document à classer. C'est là

une version de base de l'algorithme que l'on peut raffiner. Souvent, on pondère les voisins par la distance qui les sépare du nouveau texte [04].

3.2.3. Réseaux de neurones :

Un **réseau de neurones** (ou *Artificial Neural Network* en anglais) est un modèle de calcul dont la conception est très schématiquement inspiré du fonctionnement de vrais neurones (humains ou non). Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type statistique grâce à leur capacité de **classification** et de **généralisation**, tels que la classification automatique de codes postaux ou la prise de décision concernant un achat boursier en fonction de l'évolution des cours. Ils enrichissent avec un ensemble de paradigmes permettant de générer de vastes espaces fonctionnels, souples et partiellement structurés. Ils appartiennent d'autre part à la famille des méthodes de l'intelligence artificielle qu'ils enrichissent en permettant de prendre des décisions s'appuyant davantage sur la perception que sur le raisonnement logique formel [04].

Exemple :

Une banque peut générer un jeu de données sur les clients qui ont effectué un emprunt constituées : de leur revenu, de leur âge, du nombre d'enfants à charge... et s'il s'agit d'un bon client. Si ce jeu de données est suffisamment grand, il peut être utilisé pour l'entraînement d'un réseau de neurones. La banque pourra alors présenter les caractéristiques d'un potentiel nouveau client, et le réseau répondra s'il sera bon client ou non, en généralisant à partir des cas qu'il connaît [04].

Structure du réseau :

Un réseau de neurone est en général composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. Chaque couche (i) est composée de N_i neurones, prenant leurs entrées sur les N_{i-1} neurones de la couche précédente. À chaque synapse est associé un poids synaptique, de sorte que les N_{i-1} sont multipliés par ce poids, puis additionnés par les neurones de niveau i, ce qui est équivalent à multiplier le vecteur d'entrée par une matrice de transformation. Mettre l'une derrière l'autre, les différentes couches d'un réseau de neurones reviendrait à mettre en cascade plusieurs matrices de transformation et pourrait se ramener à une seule matrice, produit des autres, s'il n'y avait à chaque couche, la fonction de sortie qui introduit une non linéarité à chaque étape. Ceci montre l'importance du choix judicieux d'une bonne fonction de sortie : un réseau de neurones dont les sorties seraient linéaires n'aurait aucun intérêt [23].

Au delà de cette structure simple, le réseau de neurones peut également contenir des boucles qui en changent radicalement les possibilités mais aussi la complexité. De la même façon que des boucles peuvent transformer une logique combinatoire en logique séquentielle, les boucles dans un

réseau de neurones transforment un simple dispositif de reconnaissance d'inputs en une machine complexe capable de toutes sortes de comportements [23].

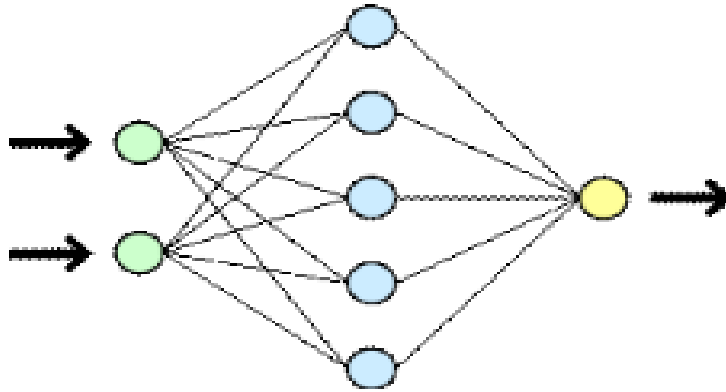


Figure 3.1 : Vue simplifiée d'un réseau artificiel de neurones [04].

3.2.4. Machines à support vectoriel :

Les machines à support vectoriel («*Support Vector Machines*» ou **SVM**) proposée dans (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 classe 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. (Figure 3.3) [03].

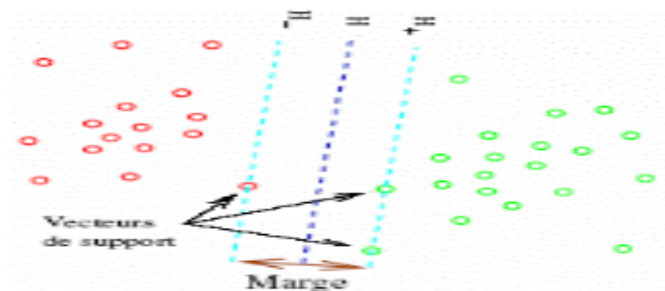


Figure 3.2 : Maximisation de la marge avec les SVM [04].

○ : les individus négatif .

○ : les individus positifs .

Des techniques de programmation quadratique permettent de résoudre ce problème. Une propriété intéressante des SVM est que la surface de décision est déterminée uniquement par les points qui en sont les plus proches de vecteurs de support. En présence de ces seuls exemples d'entraînement, la même fonction serait apprise.

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 des fonctions appelées fonctions noyaux. [03].

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. Les SVM conviennent bien pour la classification de textes. 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. Egalement, 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 [03].

3.2.5. Classifieur bayésien naïf :

Comme son nom l'indique, ce classificateur se base sur le théorème de Bayes permettant de calculer les probabilités conditionnelles. Dans un contexte général, ce théorème fournit une façon de calculer la probabilité conditionnelle d'une cause sachant la présence d'un effet, à partir de la probabilité conditionnelle de l'effet sachant la présence de la cause ainsi que des probabilités a priori de la cause et de l'effet.

On peut résumer son utilisation lorsqu'il est appliqué à la classification de textes ainsi :

- On cherche la classification qui maximise la probabilité d'observer les mots du document.
- Lors de la phase d'entraînement, le classificateur calcule les probabilités qu'un nouveau document appartienne à telle catégorie à partir de la proportion des documents d'entraînement appartenant à cette catégorie. Il calcule aussi la probabilité qu'un mot donné soit présent dans un texte, sachant que ce texte appartient à telle catégorie.

- ✓ Par la suite, quant un nouveau document doit être classé, on calcule les probabilités qu'il appartienne à chacune des catégories à l'aide de la règle de Bayes et des chiffres calculés à l'étape précédente.

La probabilité à estimer est donc : $P(C_j | a_1, a_2, a_3, \dots, a_4)$

Où:

- C_j est une catégorie
- a_i est un attribut

A l'aide du théorème de Bayes, on obtient [04]:

$$P\left(a_1, a_2, a_3, \dots, \frac{a_4}{C_j}\right) * \left(\frac{P(C_j)}{(a_1, a_2, a_3, \dots, a_4)}\right) \quad (4)$$

La probabilité qu'un mot apparaisse dans un texte est indépendante de la présence des autres mots du texte. On sait que cela est faux. Par exemple, la probabilité de présence du mot «artificielle» dépend partiellement de la présence du mot «intelligence». Pourtant, cette supposition n'empêche pas un tel classificateur de présenter des résultats satisfaisants. Et surtout, elle réduit de beaucoup les calculs nécessaires. Sans elle, il faudrait tenir compte de toutes les combinaisons possibles de mots dans un texte, ce qui d'une part impliquerait un nombre important de calculs, mais aussi réduirait la qualité statistique de l'estimation, puisque la fréquence d'apparition de chacune des combinaisons serait très inférieure à la fréquence d'apparition des mots pris séparément. [03].

Pour estimer la probabilité $P(a_i | C_j)$, on pourrait calculer directement dans les documents d'entraînement la proportion de ceux appartenant à la classe c_j qui contiennent le mot a_i . Cependant, l'estimation ne serait pas très valide pour des numérateurs petits.

Dans le cas extrême où un mot ne serait pas du tout rencontré dans une classe, sa probabilité de 0 dominerait les autres dans le produit ci-dessus et rendrait nulle la probabilité globale. Pour pallier ce problème, une bonne façon de faire est d'utiliser le m-estimé qui est calculé ainsi :

$$\frac{n_k + 1}{n + |\text{vocabulaire}|} \quad (5)$$

Où

- n_k : est le nombre d'occurrences du terme dans la classe c_j .
- n : est le compte total des mots dans le corpus d'entraînement.
- $|\text{Vocabulaire}|$: Le nombre de mots clés [04].

Avantages et inconvénients :

Avantage :

- Possible en ligne.

– Complexité acceptable.

Inconvénients :

– Génératif : les modèles génératifs fournissent de bons estimateurs lorsque le modèle est "juste", ou en terme statistique bien spécifié, ce qui signifie que le processus qui génère les données réelles induit une distribution qui est celle du modèle génératif. Lorsque le modèle est mal spécifié (ce qui est de loin le cas le plus courant) on aura intérêt à utiliser une méthode discriminative. [24]

3.3. Evaluation des performances d'un classifieur :

Pour évaluer les résultats obtenus par un classifieur, les documents de l'espace d'apprentissage sont souvent divisés en deux ensembles : le premier est utilisé pour la construction du classifieur tandis que le deuxième est utilisé pour faire le test. Puisqu'on adopte l'approche de classification supervisée, on connaît à l'avance la catégorie de chaque document. Ainsi, on compare la catégorie prédite avec celle prédéfinie et on calcule un score de performance. Ce calcul peut se faire de diverses façons. Dans ce qui suit, nous allons présenter les méthodes qu'on a utilisé pour mesurer la performance des classifieur et telles qu'elles étaient présentées par [14]. Il faut noter que ces méthodes sont souvent utilisées dans la littérature.

Les paramètres d'évaluation standard sont : la précision (P) et le rappel (R).

➤ Définitions :

La précision et le rappel sont deux quantités qui sont définies lorsque les filtres prennent des décisions binaires : soit un document est sélectionné par le filtre, soit il ne l'est pas. Lorsque les ensembles de documents pertinents et non pertinents sont connus sur un corpus, il est alors possible d'évaluer les quantités définies à la Figure 4.1 [25].

	Pertinents	Non pertinents
Sélectionnés	a	b
Non sélectionnés	c	d
Total	P	NP

Figure 3.1 : Table de contingences pour un filtre binaire.

Le rappel R et la précision P et sont définis par :

$$R = a / a + c$$

$$P = a / a + b$$

Le *rappel* est le rapport du nombre de documents pertinents trouvés par le filtre au nombre de documents pertinents disponibles. Il s'agit de la proportion de documents bien classés pour la classe des documents pertinents : c'est une mesure utilisée habituellement en classification.

La *précision* est la proportion de documents pertinents parmi les documents sélectionnés.

Cette quantité ne représente pas un taux d'exemples bien classés par rapport à une classe et n'est donc pas normalisée.

Ces deux notions sont souvent utilisées, car elles reflètent le point de vue de l'utilisateur : si la précision est faible, l'utilisateur sera insatisfait, car il devra perdre du temps à lire des informations qui ne l'intéressent pas. Si le rappel est faible, l'utilisateur n'aura pas accès à une information qu'il souhaitait avoir [25].

Un filtre parfait doit avoir une précision et un rappel de un, mais ces deux exigences sont souvent contradictoires et une très forte précision ne peut être obtenue qu'au prix d'un rappel faible et vice-versa.

En effet, dans le cas limite où aucun document n'est sélectionné, la précision vaut un et le rappel est nul. Dans le cas limite où tous les documents sont sélectionnés, le rappel vaut un et la précision est de $P / (P + NP)$; cette quantité, appelée *densité* du thème, est généralement assez faible, et représente la précision moyenne que l'on obtiendrait en sélectionnant les documents aléatoirement.

Dans la pratique, les valeurs exactes de la précision et du rappel ne peuvent pas être calculées et, par conséquent, les valeurs absolues n'ont pas de sens.

Pour mesurer les performances d'un filtre (comme celles de tout classifieur), il faut utiliser une base de test indépendante de la base d'apprentissage. Pour que les résultats obtenus sur cette base de test aient un sens, deux conditions doivent être remplies : il faut, d'une part, que cette base soit suffisamment grande et représentative, et, d'autre part, il faut pouvoir évaluer les quantités a , b , c et d sur cette base. Or la connaissance de ces quantités nécessite la catégorisation manuelle de chaque document de la base, ce qui est justement très difficile à faire si la base est grande [25].

➤ On peut distinguer aussi autre mesure de performance :

- ✓ exactitude «accuracy» : $(a + d) / (a + b + c + d)$.
- ✓ erreur «error» : $(b + c) / (a + b + c + d)$.

Les deux dernières mesures, bien que couramment utilisées en apprentissage automatique, sont jugées moins bien adaptées à la tâche de classification de textes. Elles incluent dans leur définition le nombre total de documents. Or, comme un document appartient généralement à un petit nombre de catégories sur l'ensemble, un classifieur qui rejeterait tous les documents présenterait seulement un faible taux d'erreur et une exactitude quand même très élevée. Entraîner un classifieur sur la base de l'optimisation d'un de ces deux critères tendrait à créer un programme qui n'accepte aucun

document dans sa catégorie. C'est la raison pour laquelle la précision et le rappel sont les mesures les plus rencontrées dans la littérature [26].

Ce qui est recherché, c'est un score global et non un score pour chaque catégorie. Il y a deux principales façons de faire une moyenne pour obtenir un score global. La macro-moyenne calcule d'abord les scores pour chaque catégorie et fait ensuite une moyenne sur ces scores. La micro-moyenne regroupe d'abord les données de chaque catégorie dans une même table de contingence globale et calcule ensuite les scores à partir de celle-ci. La distinction à faire entre ces deux méthodes est que la macro-moyenne donne une importance égale à toutes les catégories, tandis que la micro-moyenne donne une importance égale à tous les documents. Les avis sont partagés quant à savoir laquelle des deux mesures est préférable. Les tenants de la macro-moyenne affirment que la micro-moyenne est moins représentative parce que les catégories plus fréquentes ont plus de poids que les autres. Et c'est justement cet aspect de la micro-moyenne que ses défenseurs apprécient [27]. Bref, il semble préférable, lors de la présentation des résultats d'une expérience, d'identifier clairement la mesure utilisée. La comparaison avec d'autres expériences sera ainsi plus facile.

Lors de l'évaluation de la performance d'un classifieur, on ne peut tenir compte de la précision ou du rappel séparément. Effectivement, on pourrait mettre en place un système qui rejeterait tous les textes : il obtiendrait une précision de 100%, mais un rappel de 0%. À l'inverse, un système qui accepterait tous les textes aurait un rappel de 100%, mais une précision de 0%. On voit donc que ce vers quoi il faut tendre est un classifieur qui fait le compromis idéal entre ces deux facteurs [26].

Dans cet ordre d'idées, on peut calculer le «break-even point», c'est-à-dire le point où la précision et le rappel sont égaux. On a souvent utilisé cette mesure pour comparer la performance de classifieur. Plus ce point se rapproche de 100%, plus le classifieur est performant à la fois en précision et en rappel.

Aussi, la mesure F1 est beaucoup utilisée. Elle est définie ainsi :

$$F = \frac{2rp}{r + p}$$

où

- r est le rappel.
- p est la précision.

C'est une fonction qui est maximisée quand la précision et le rappel sont proches. On cherche généralement à l'optimiser lors de l'ajustement du seuil. On peut aussi utiliser une forme plus générale de la mesure F en ajoutant un paramètre qui pondère l'importance relative des deux critères. Il se peut en effet qu'une application particulière nécessite une précision élevée, mais puisse se permettre un rappel un peu moins bon et vice versa [28].

À la vue de cette variété de critères, un des points importants à considérer lorsqu'il s'agit d'évaluer en parallèle différents classifieurs est la comparabilité des mesures de performance utilisées. Chaque chercheur doit donc se faire un devoir de mentionner précisément la façon dont il a procédé pour évaluer son classifieur. Ainsi, la comparaison de différentes techniques de classification automatique, qui apparaît déjà assez complexe, serait un peu plus facile [26].

3.4. Quelle est la meilleure méthode pour la catégorisation de textes ?

Comme beaucoup d'approches différentes ont été utilisées pour la catégorisation de textes, une des questions récurrentes est : quelle est la meilleure méthode pour la catégorisation de textes ?

Il existe, en pratique, plusieurs méthodologies pour tenter de répondre à cette question ; nous les décrivons ci-dessous.

La première consiste à comparer différentes méthodes mises en œuvre par différents auteurs sur le même corpus. L'inconvénient de cette méthode est qu'il faut que tous les auteurs utilisent exactement le même découpage du corpus. Pour le corpus Reuters-21578, qui est souvent utilisé, certains auteurs considèrent 90 catégories (Joachims, 1998), (Schapire *et al.*, 1998), (Yang et Liu, 1999), d'autres en considèrent 118 (Dumais *et al.*, 1998). De plus, la plupart des auteurs considèrent 3299 documents sur la base de test, mais (Yang et Liu, 1999) en considèrent uniquement 3019 en supprimant tous les documents de la base de test qui n'appartiennent à aucune catégorie [25].

Finalement, ces légères différences de découpage rendent difficiles les comparaisons à travers ces publications. De plus, tous les auteurs n'utilisent pas les mêmes mesures de performances, et peuvent calculer les moyennes de manières différentes (les différentes mesures sont présentées au chapitre 4). Enfin, même dans le cas où les auteurs utilisent les mêmes mesures, il est nécessaire d'utiliser des tests statistiques pour vérifier que les différences ne sont pas dues au hasard (Hull, 1993).

Une autre approche souvent proposée est l'utilisation de plusieurs méthodes par le même auteur ; de cette manière, le découpage et les mesures sont identiques pour toutes les méthodes.

(Yang et Liu, 1999) comparent ainsi les machines à vecteurs supports, les plus proches voisins, les réseaux de neurones, une combinaison linéaire, et des réseaux bayésien. (Dumais *et al.*, 1998) proposent également une série de comparaisons en mettant en compétition une variante de l'algorithme de Rocchio (appelée *find similar*), des arbres de décision, des réseaux bayésien et des machines à vecteurs supports [25].

Le problème vient du fait que toutes ces méthodes sont délicates à mettre en œuvre et leurs performances dépendent fortement des algorithmes utilisés.

Par exemple, l'implémentation des machines à vecteurs supports proposées par (Dumais *et al.*, 1998) obtient de nettement meilleurs résultats que celle proposée par (Joachims, 1998).

Les réseaux de neurones testés par (Yang et Liu, 1999) sont des perceptrons multi-couche avec une couche cachée comportant 64 neurones, 1000 descripteurs en entrées et 90 neurones de sorties correspondant aux 90 catégories ; ils considèrent un seul réseau pour l'ensemble des catégories comportant plus de 64000 poids (l'algorithme d'apprentissage n'est pas précisé). Il n'est pas surprenant, dans ces conditions, que les performances obtenues ne soient pas très bonnes : de telles démarches jugent plus la capacité des auteurs à mettre en œuvre des méthodes, que les capacités des méthodes elles-mêmes [25].

L'algorithme de Rocchio est considéré comme un algorithme ancien, mais (Schapire *et al.*, 1998) ont montré que cet algorithme obtient d'excellents résultats pour la catégorisation de textes à condition d'utiliser un codage efficace, de bien choisir les documents non pertinents, et d'effectuer une optimisation des poids ("*a state of the art version of Rocchio's algorithm is quite competitive with modern machine learning algorithms for text filtering*"). Leurs conclusions vont à l'encontre d'autres comparaisons qui montrent que cet algorithme n'est pas performant par rapport aux méthodes fondées sur l'apprentissage numérique (Schütze *et al.*, 1995)(Lewis *et al.*, 1996)(Cohen et Singer, 1996)[25].

Ces différentes remarques prouvent que le succès d'une méthode dépend d'un ensemble de paramètres qui vont du codage des documents au choix des algorithmes et de leur utilisation, et qu'il est, par conséquent, extrêmement difficile de tirer des conclusions définitives sur une approche.

3.5. Conclusion :

La classification supervisée de documents a fait beaucoup de progrès ces dernières années. Nous avons présenté dans ce chapitre quelques techniques de la catégorisation automatique de texte et nous basons dans notre travail sur la méthode Naïve Bayes et la méthode arbre de décision que nous allons présenter dans le prochaine chapitre avec plus de détaille.