Chapitre 3

Approches de classification : Etat de l'art

Table des matières

3.1- Introduction

Avant de présenter les approches les plus utilisées dans la littérature, en exposant les principes de base, les particularités de chaque méthode de classification de textes (section 3.3), nous allons introduire ce chapitre par un petit rappel sur le principe de base de l'apprentissage automatique et particulièrement le supervisé, après nous pourrons conclure facilement que la problématique de catégorisation de textes se situe bien dans le cadre de l'apprentissage supervisé, et pour en finir cette introduction, nous donnerons un aperçu sur la façon avec laquelle les documents sont classés, le chapitre qui va suivre définit les formules les plus connues pour le calcul de distance qui va être utilisée dans plusieurs méthodes de classifications comme les SVM, kPPV ou Rocchio .

3.1.1- L'apprentissage automatique

Puisque l'approche manuelle de classification de textes est coûteuse en temps de travail, peu générique, et relativement peu efficace, l'autre solution a été admise, qui consiste à faire apprendre automatiquement à l'ordinateur, sur la base d'un corpus de textes qui servent d'exemples, les paramètres de la fonction de classement.
Ainsi depuis une quinzaine d'années la classification de textes a été considérée comme un problème d'apprentissage automatique et est rapidement devenue un champ d'essai sollicité par les différentes techniques de classification.
De toute façon, quelle que soit l'approche retenue, une des particularités de cette tâche est la très grande dimensionnalité de l'espace dans lequel les textes sont représentés, qui comprend généralement plusieurs milliers de termes.
L'apprentissage automatique s'intéresse aux méthodes inductives permettant d'acquérir des connaissances à partir d'observations d'un phénomène. Cette connaissance peut être exploitée pour des tâches de décision ou de prévision : c'est le cadre de l'apprentissage supervisé ; ou à des fins d'analyse exploratoire ou de structuration d'un ensemble de données : c'est le cadre de l'apprentissage non-supervisé (Yvon, 2006).
Le contexte de notre étude se situe dans le premier cas.

3.1.2- L'apprentissage supervisé

Le cadre général de l'apprentissage supervisé consiste, à partir de l'observation d'un ensemble de couple de données de la forme [(x(i), y(i)], (i = 1 ---> n), à induire la valeur de y pour de nouvelles valeurs de x. Dans un cadre probabiliste, chaque x(i) représente une observation d'une variable aléatoire X.
Suivant les valeurs de la variable aléatoire Y, deux cas de figures peuvent être distingués : lorsqu'elle prend des valeurs discrètes, on parle de catégorisation, lorsque ses valeurs sont discrètes, de régression.
Le cadre statistique de la catégorisation supervisée considère le problème de l'apprentissage comme celui de l'induction d'une fonction f. Sous l'hypothèse que les observations sont indépendantes et uniformément distribuées selon une loi de probabilité inconnue P, la meilleure fonction (hypothèse) sera celle qui a une espérance de risque minimum (Yvon, 2006).
Ce cadre général est bien connu et plusieurs outils statistiques ont été utilisés dans le domaine pour résoudre ce problème : réseaux de neurones, régression logistique, Formule de Bayes, arbres de décisions, k-plus proche voisins, machines à vecteurs de support (SVMs), boosting, et bien d'autres. Ces modèles se généralisent plus ou moins directement aux situations dans lesquelles Y prend plus de deux valeurs (catégorisation multiclasses).

3.1.3- La catégorisation est un problème de classification supervisée

Pour construire un filtre relatif à une classe donnée, il faut donc disposer de couples (Document, Classe), ces exemples de chaque classe, préalablement étiquetés constituent le corpus d'apprentissage.
On fait appel aux méthodes d'apprentissage supervisées pour ajuster un modèle qui crée une association entre les documents d'entrée et les classes de sortie. Ainsi, par ces méthodes d'apprentissage, il est possible de construire un modèle de classification, à partir de ces exemples connus à priori (Document, Classe). (Jalam, 2003)
Ce qui affirme clairement que la catégorisation de textes est bien un problème de classification supervisée.

3.1.4- Comment classer ?

Classer les documents revient en réalité à déterminer les paramètres de la fonction de classement. Voici l'idée globale de ce qu'on doit faire :

- Il faut disposer d'un corpus d'apprentissage, qui va servir d'entrée à un algorithme d'apprentissage.
- On sélectionne un autre corpus qui sert pour l'évaluation (corpus de test)
- Il faut d'abord déterminer les descripteurs (variables de la fonction)
- Il faut fournir à l'ordinateur un type de fonctions de classement lui permettant d'associer une catégorie à un texte.
    - SVMs
    - Naïve Bayes
    - Règles de décision
    - Arbres de décision
    - Réseaux de neurones
    - Autres fonctions ...
- On infère, à partir des données, et par des méthodes mathématiques complexes, les paramètres de la fonction de classement utilisé, qui peuvent être :
    - Coefficients de l'hyperplan dans les SVMs
    - Distributions de probabilité dans les classificateurs probabilistes
    - Règles dans les règles de décision
    - Conditions et branchements dans les arbres de décision
    - Poids dans les réseaux de neurones
    - …
- On se fonde sur la connaissance préalable des bonnes catégories pour les documents du corpus d'apprentissage (apprentissage supervisé).

3.2- Différents modèles de classifieurs

Historiquement, différentes générations d'algorithmes de classification automatique de textes se sont succédé. Une part d'amélioration a été apportée par chaque nouvelle génération par rapport aux antécédentes. Parmi les premières, nous trouverons certainement les approches sémantiques dont l'handicap principal résidait dans le coût excessif puisque plusieurs experts sont chargés dans des intervalles de temps importants, pour mettre à jour les plans de classement.
Pour apaiser à ces limitations, plusieurs boites ont conçues et commercialisées des technologies de catégorisation automatique basées sur des approches purement statistiques.

Plusieurs générations de techniques statistiques ont depuis été développées et qui ont confirmé leurs performances en obtenant de meilleurs résultats.

Actuellement, plusieurs approches de catégorisation coexistent. Nous citons parmi les plus utilisées le modèle probabiliste Naïve Bayes, ou les modèles vectoriels de Machine à Vecteur de Support, les k-plus-proches voisins, Rocchio, ainsi que des modèles à base de règles ou d'arbre de décision ou encore des approches fondées sur les réseaux de neurones qui ont été proposées. Chacun de ces modèles possède certains avantages et certains inconvénients. Dans ce qui suit une liste plus ou moins exhaustive des différents modèles et de leur intérêt respectif sera présentée. Une description plus détaillée sera accordée à l'approche naïve bayesienne dans le chapitre 6 qui fera l'objet des modèles de classification dans notre approche proposée.

3.2.1- Machines à Vecteurs Support – SVM

3.2.1.1- Présentation de l'approche

L'algorithme SVM (Support Vector Machine) est une méthode d'apprentissage supervisée relativement récente introduite pour résoudre un problème de reconnaissance de formes à deux classes (Vapnik, 1995). Le principe de SVM a été proposé par Vapnik à partir de la théorie du risque empirique.
La méthode SVM est un classificateur linéaire utilisant des mesures de distance.
En ce qui concerne son application à la problématique de catégorisation de documents, l'algorithme repose sur une interprétation géométrique simple est l'idée générale est de représenter l'espace des exemples (ici des documents) dans un espace vectoriel où chaque document étant un point dans cet espace et de trouver la meilleure séparation possible de cet espace en deux classes. L'espace de séparation est une surface de décision appelée marge, défini par les points « vecteur support ». Ces points se trouvent au minimum de marge. La marge se présente alors comme la plus courte distance entre un vecteur de support et "son" hyperplan. La marge se définit comme la plus petite distance entre les exemples de chaque classe et la surface séparatrice S :

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

Ainsi la décision s'appuie sur les SVM pour couper l'espace en deux : d'un côté, ce qui est dans la catégorie, de l'autre côté, ce qui n'y est pas.
l'approche par SVM permet donc de définir, par apprentissage, un hyperplan dans un espace vectoriel qui sépare au mieux les données de l'ensemble d'apprentissage en deux classes, minimisant le risque d'erreur et maximisant la marge entre deux classes. La qualité de l'hyperplan est déterminée par son écart avec les hyperplans parallèles les plus proches des points de chaque classe. Le meilleur hyperplan est celui qui a la marge la plus importante.
SVM a été étendu pour les points ne pouvant être séparées de manière linéaire (par exemple notre cas des vecteurs de documents), en transformant l'espace initial des vecteurs de données à un espace de dimension supérieure dans lequel les points deviennent séparables linéairement. Nous trouvons dans (Joachims, 1998) une application efficace des SVM.
La figure 3.1 montre une telle séparation dans le cas d'une séparation linéaire par un hyperplan.

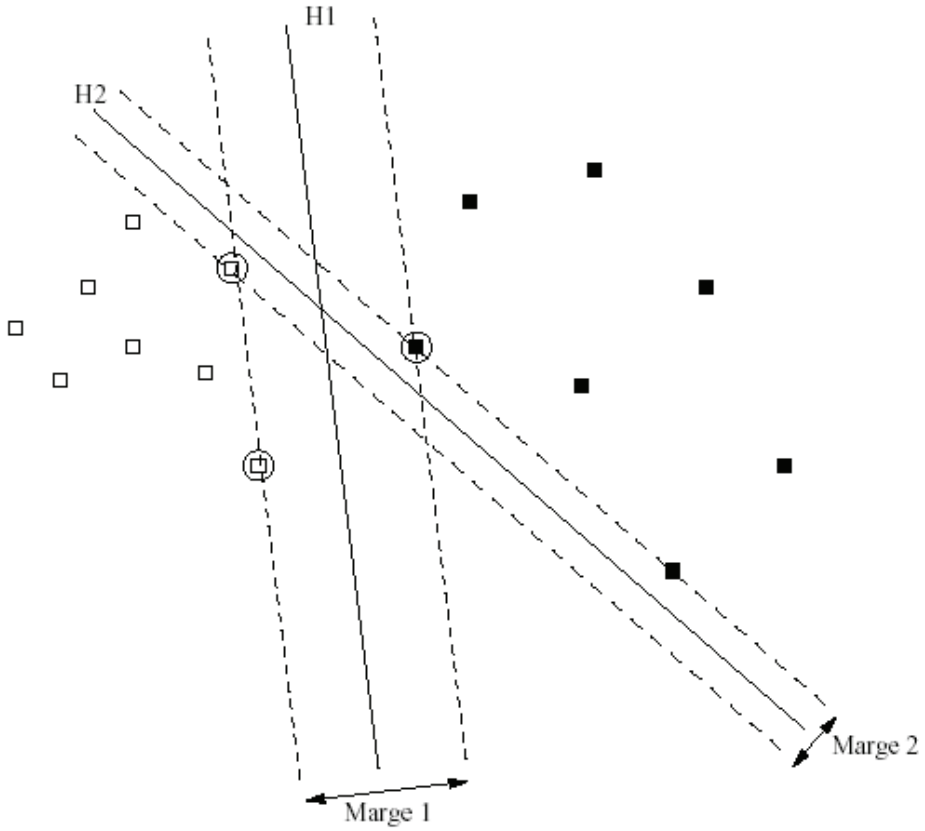

Figure 3.1: Exemples d'hyperplans séparateurs en dimension deux. Les vecteurs de support sont encerclés

Dans l'exemple de la figure 3.1, 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 (w,b) (b est la distance à l'origine de l'hyperplan) qui minimise la norme de w sous les contraintes :

$$\forall d_i,\ c_i(w.d_i - b) \leq 1$$

Avec d_i le i^{éme} document, de classe c_i (+1 ou –1). Si les exemples ne sont pas linéairement séparable, on peut les plonger conceptuellement dans un espace de dimension plus grande (la dimension peut même être infinie) par une fonction de transformation appelée *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 calculable 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 multi-classe). La méthode la plus commune pour résoudre un problème multi-classe reste de le transformer préalablement en plusieurs sous-problèmes bi-classe.
Cet algorithme est particulièrement bien adapté à la catégorisation de textes car il est capable de gérer des vecteurs de grande dimension. Dans la pratique, 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 dans le domaine pour la première fois par Joachims qui a notamment travaillé à rendre SVM compatible avec les données textuelles qui sont caractérisées par de grandes dimensions avec des matrices (documents * termes) très creuses.
Depuis, l'approche a été très souvent réutilisé, par exemple pour la détection de courriers électroniques non sollicités ou pour la classification de dépêches.

3.2.1.2- Critiques de l'approche

Actuellement, l'algorithme SVM semble tres Prometteuse et considéré parmi les plus performants pour la catégorisation en raison de sa modélisation simpliste et rapide à calculer par une machine étant donné que SVM est un Classificateur linéaire qui correspond à une équation polynomiale de degré p :

$a_{1x}\,x+a_{1y}\,y+a_{1z}\,z$ pour la catégorie C1

$a_{2x}\,x+a_{2y}\,y+a_{2z}\,z$ pour la catégorie C2

Seulement elle introduit des concepts complexes peu adaptés aux corpus de grandes tailles non fixes, sans oublier de rappeler de son faible pouvoir descriptif puisque les coefficients ne sont pas interprétables intuitivement par des humains.

3.2.2- Rocchio

3.2.2.1- Présentation de l'approche

La méthode Rocchio parue dans (Rocchio, 1971), est un classifieur linéaire basée sur le calcul des mesures de distance, il fait partie des premières techniques de classification supervisée. Plusieurs améliorations se sont apportés sur le modèle mais la définition présentée ici est celle de la version initiale.

Dans la phase d'apprentissage de la méthode, les représentations vectorielles, vont permettre le codage de chaque catégorie par un vecteur dont lequel figure tout les termes générés avec leur nombre d'occurrence. Le processus représente donc les classes par des profils prototypiques correspondants à des vecteurs dans un espace vectoriel similaire aux documents. Ces profils sont donc des listes de termes pondérés générées pendant l'apprentissage de même pour les vecteurs correspondants aux textes qui sont aussi générés durant cette phase. Le profil d'une catégorie doit contenir tous les termes qui caractèrisent cette catégorie par rapport aux autres.
Bien évidemment, le vecteur d'une classe sera calculé ici uniquement en fonction des documents du jeu d'apprentissage. Pour construire les profils, il est nécessaire d'utiliser une méthode de sélection et réduction de termes.
Chaque terme du corpus représente une dimension de l'espace et le codage des vecteurs se fait soit par une fonction booléenne soit par une fonction du nombre d'occurrences d'un terme dans le document. Plusieurs pondérations des termes se présentent, la plus utilisée est TFIDF.

Dans la phase de classification, il s'agit de comparer le profil (vecteur) du nouveau document à classer à tous les profils des classes déjà calculés dans l'étape d'apprentissage.
Cette comparaison équivaut au calcul d'une fonction de similarité ou de distance entre les vecteurs représentant les classes et le vecteur correspondant au nouveau document. Elle permet d'ordonner les classes en fonction de leur distance du document. Par conséquent, le principe de catégorisation Rocchio se résume à assigner le document à la classe dont la distance euclidienne entre le vecteur du document et le vecteur de la classe est la plus courte.

3.2.2.2- Critiques de l'approche

Rocchio est caractérisée par sa modélisation simpliste et rapide à calculer par une machine et malgré l'ancienneté et la simplicité du modèle, la méthode a confirmé par ses performances qui sont concurrentielles à celles obtenues par d'autres techniques plus sophistiquées, avec un gain de temps d'apprentissage pour une machine très considérable.
Plusieurs auteurs dans leurs travaux ont démontré la résistance de Rocchio au bruit : même avec 50% des exemples bruités, les performances sont assurées. Les différentes expérimentations basées cette approche ont donné aussi de très bons résultats sur les tâches de routage (filtre anti-spams par exemple). (Vinot & Yvon, 2002).
En revanche, l'inconvénient principal de la méthode reste sa faiblesse d'expressivité et son pouvoir descriptif très réduit qui fait du modèle une fonction de décision non interprétable intuitivement par les humains.

3.2.3- Méthode du centroïde

3.2.3.1- Présentation de l'approche

L'algorithme du centre de gravite ou « centroïde » est une variante de l'algorithme Rocchio bien connu dans le domaine, c'est une méthode géométrique utilisant les mesures de distance pour classer les documents.
La phase d'apprentissage consiste à calculer un centroïde pour représenter les classes. Le vecteur centroïde d'une catégorie sera représenté par la moyenne des vecteurs des textes qu'elle contient, qui correspond au barycentre des différents textes préalablement assignés à cette classe. La méthode du centroïde calcule la distance entre les centres de gravité (au sens géométrique) des catégories.

La catégorisation d'un nouveau document se fait par le calcul de similarité entre les vecteurs centroïdes des catégories et le vecteur du nouveau document. La catégorie du centroïde le plus proche est attribuée dans le cas de classification multi-classes disjointes sinon des scores sont calculés pour les classes puis un seuillage est utilisé pour choisir les classes à attribuer.

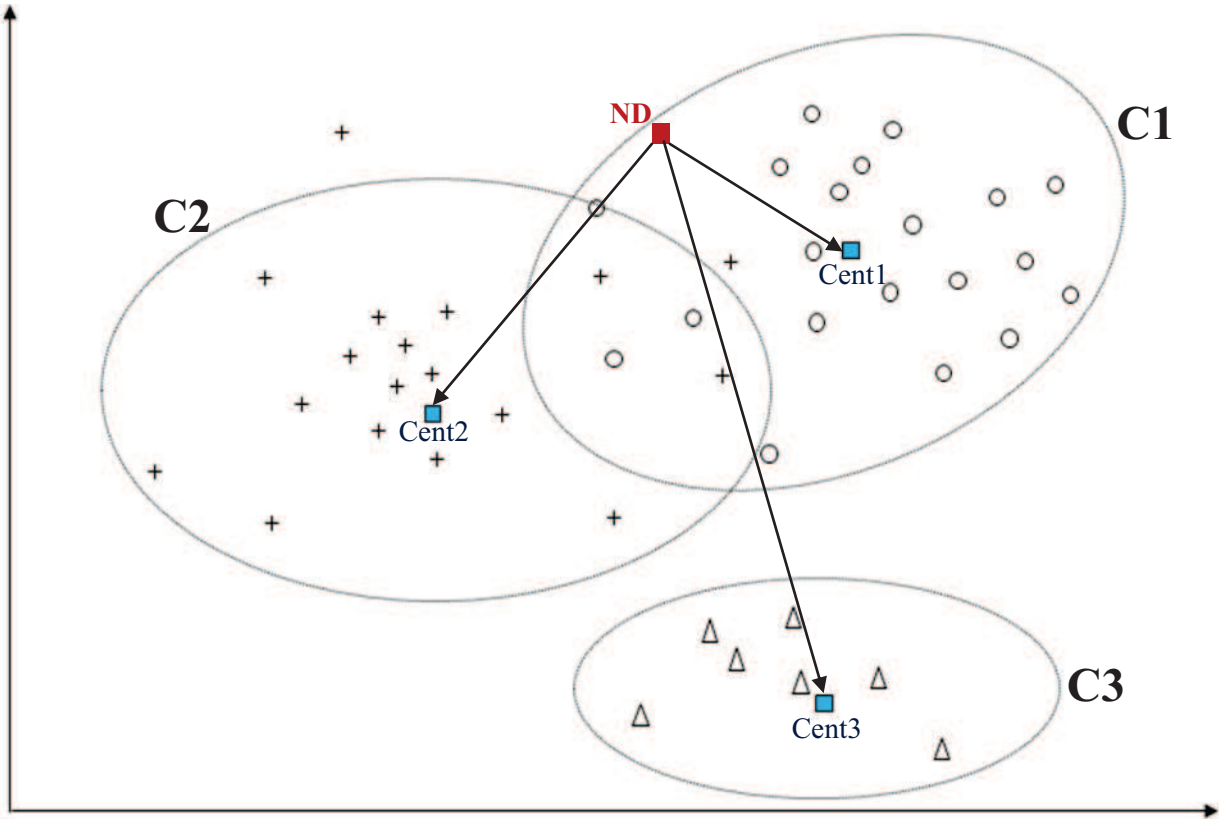

Figure 3.2: Exemple de la méthode du centroïde
Le nouveau document ND est attribué à la classe C1 du centroïde Cent1 le plus proche

3.2.3.2- Critiques de l'approche

Comme la méthode considère un vecteur représentatif pour chaque classe, elle ne fonctionne que si cette classe est bien définit par un prototype. Mais, puisque les comparaisons ne sont faites qu'avec les vecteurs centroïdes, l'algorithme est plus efficace que k-PPV (Section 3.2.4) pour la catégorisation en temps réel.
La méthode de la comparaison directe ou méthode du centroïde est caractérisée par son efficacité (rapidité), sa robustesse et son interprètabilité, en revanche son optimisation est difficile dans le cas de nombreuses classes.
Elle est très utilisée dans le domaine du traitement du Signal et d'image. Elle consiste à calculer le centre de gravité correspondant des spectres ou des pixels pour divers traitements comme la segmentation par exemple.

3.2.4- K plus proches voisins - kPPV

3.2.4.1- Présentation de l'approche

La méthode des k plus proches voisins ou The k-NN classification (k-Nearest Neighbors) est un classifieur à base d'instances qui fait partie des méthodes géométriques utilisant des mesures de distance.
k-NN est un algorithme classique d'apprentissage qui a été longtemps à la base des algorithmes de catégorisation des documents, elle a été employée avec succès dans de domaine de classification et a engendré toute une famille de classifieurs connus sous le nom de classifieurs paresseux (lazy learns). Dans ces systèmes, le seul traitement effectué au cours de la phase d'apprentissage est la mémorisation des exemples sous une forme optimale de

façon à pouvoir les extraire ensuite rapidement. Chaque texte est représenté dans un espace vectoriel, dont chacun des axes représente un descripteur. Tous les calculs sont reportés à la phase de classification (d'où le terme de paresseux).
k-NN est un algorithme de catégorisation dans lequel les classes ne sont pas représentées sous forme de texte "prototype" (profil de catégorie).
La partie classification est en contrepartie plus coûteuse en temps : Le classifieur calcule la similarité du nouveau texte à catégoriser avec l'ensemble des autres exemples du corpus d'apprentissage, dont les catégories sont déjà connues, puis il sélectionne les k documents les plus proches du document à classer. Ensuite, pour affecter la catégorie, les relations entre ces k documents et les catégories sont évaluées et un score est calculé par catégorie afin d'évaluer la pertinence de la catégorie au document. La catégorie (ou les catégories) ayant le score le plus élevé (celle qui contient le plus de textes voisins) est affectée au document (Yang, 2001).
Voici son algorithme général :

Paramètres : le nombre k de voisins
Données : un ensemble d'exemples classés (document, classe)

Entrée : un nouveau document D
1. déterminer les k plus proches documents de D
2. Sélectionner la classe majoritaire C des classes de ces k exemples

Sortie : la classe de D est C

Traduit en langage naturel: on va affecter au nouvel élément à traiter la classe la plus représentée dans ses k plus proches voisins. On cherche les « k plus proches voisins » dans les documents déjà répertoriés du nouveau document à classer. L'assignation du document peut être à plusieurs classes à la fois. Ainsi, chaque classe suffisamment représentée dépassant un certain seuil sera associé au document. Le seuil minimum acceptable pour qu'un document se voit assigner à une classe (en cas de multi-catégories) se calcul à partir des documents d'entraînements et représente une des difficultés de la mise en œuvre de cet algorithme.
Pour que cet algorithme soit efficace, il faut utiliser une bonne mesure de similarité entre les documents, notamment afin que les attributs non discriminant ne soient pas pris en compte et que ceux qui sont discriminants 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 convient donc parfaitement et les k-PPV sont bien adaptés à ce modèle.
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 à 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 où les documents sont représentés dans un espace de très grandes dimensions (le calcul de la similarité prend dans ce cas un temps non négligeable). 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 inversé (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 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.

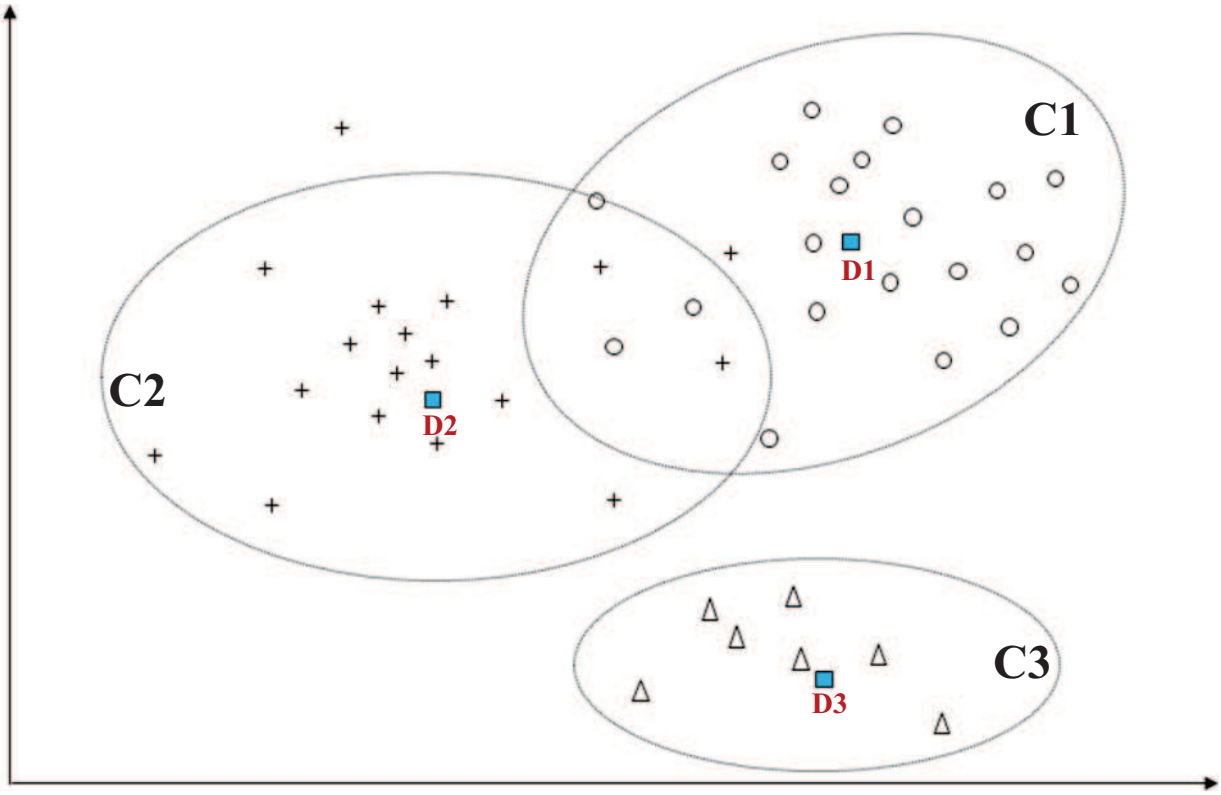

Figure 3.3 : Exemple de la méthode k-PPV
Les trois nouveaux documents D1, D2 et D3 sont associés respectivement aux classes des k plus proches voisins C1, C2 et C3.

Il existe néanmoins plusieurs versions de k-PPV. Ces versions peuvent diverger, principalement, sur la politique de classification. Pour choisir la classe la plus proche, on peut procéder de plusieurs manières : Certaines méthodes choisiront comme classe candidate pour un texte, la classe majoritairement retrouvée sur ses k plus proches documents voisins. D'autres pondèreront les classes candidates en fonction de la position du texte dans le classement (des k plus proches voisins), avant de réaliser notre choix final. Dans (Yang, 1999) nous pouvons trouver une description plus détaillée des différentes versions de k-PPV.

3.2.4.2- Critiques de l'approche

Selon les tests sur l'ensemble des techniques statistiques pratiqués par (Yang, 1999) dans le domaine de classification, kPPV est actuellement une des méthodes les plus efficaces en qualité de classement en raison de sa performance très stable puisque elle donne toujours de bons résultats tout en étant simple et facile à implémenter.
Néanmoins, Un ensemble de limitations des plus proches voisins découle, voici quelques limites qui affectent la catégorisation de textes :

1. Le principal problème que présente cette méthode est sa prédiction longue : du point de vue temps d'exécution, les algorithmes de type plus proches voisins sont longs en phase de classification, puisqu'on sauvegarde et on fait appel à tous les exemples du corpus d'apprentissage (le document à classer est comparé par calcul de similarité à l'ensemble des documents de la base). Et pour chaque nouvel élément à traiter, il faut entièrement tout recalculer, ce qui implique un coût de traitement important.

2. Par ailleurs, kPPV est sensible aux documents (exemples) négatifs dans la base d'apprentissage.
3. Fait partie des algorithmes locaux, car au moment de prise de décision on n'utilisent qu'une faible partie des exemples à chaque classification.(K plus proche voisins)
4. Les kPPV sont sensibles au choix de la fonction de similarité de l'algorithme.
5. Le principe du kPPV s'appuie sur la proximité entre les individus, néanmoins l'utilisation de ce seul critère parait insuffisante. En effet du point de vue cognitif lorsqu'un terme x a pour plus proche voisin y, on s'attend normalement que y est réciproquement a pour plus proche voisin x mais malheureusement ce n'est pas le cas puisque dans notre exmple, z qui est le plus proche voisin de y.

Figure 3.4 : Principe des kPPV

3.2.5- Arbres de décision

3.2.5.1- Présentation de l'approche

Les arbres de décision, utilisés dés les années 60 en statistique, sont des outils supervisés. Ils sont devenus, depuis une vingtaine d'années des outils très populaires pour générer des règles de classification et plus généralement des règles de prédictions. On parle ainsi d'arbre de classification lorsque la variable à prédire est catégorielle et que ses valeurs représentent donc des classes.
Les arbres de décision ont été utilisés pour permettre la catégorisation des documents dans un certain nombre de classes prédéfinies. L'apprentissage des probabilités d'attribution d'un texte à une classe est réalisé sur des textes étiquetés manuellement. L'arbre de décision généré, sur l'ensemble des documents du corpus d'apprentissage, permet de décider à quelle classe appartient chaque nouveau document du corpus de test. Chaque feuille de l'arbre contient la probabilité d'appartenance à l'une ou l'autre des classes. Suivant les réponses aux questions posées au document à classer, celui-ci est « dirigé » vers telle ou telle feuille de l'arbre. Le document est alors attribué à la classe de plus forte probabilité.(Bellot, 2000)

Selon Gilbert Ritschard, Simon Marcellin et Djamel A. Zighed dans (Ritschard & all, 2009), un partitionnement récursif des données est le principe de base sur lequel s'appuie la méthode pour avoir des ensembles cohérents par rapport à la variable à prédire. Un enchaînement hiérarchique de règles est généré. Chaque feuille (nœud terminal) de l'arbre caractérise une règle, de type « Si condition Alors résultat », dont la prémisse est définie par les conditions d'embranchement le long du chemin menant du nœud initial à la feuille, la conclusion de la règle correspondant à la classe assignée à la feuille. L'ensemble de toutes ces règles représente le modèle de prédiction.
En effet, Pour construire l'arbre de décision, il faut chercher lequel des attributs (termes) qu'il faut tester à chaque nœud, c'est un processus récursif. Pour décider de l'attribut qu'il faut tester à chaque étape, on utilise un calcul statistique qui détermine dans quelle mesure cet attribut sépare bien les exemples Oui/Non. On crée alors un nœud contenant ce test, et on crée autant de descendants que de valeurs possibles pour ce test.
Ainsi, si on teste la présence d'un mot, les valeurs possibles sont « Présent/Absent ». A chaque fois, on aura donc deux descendants pour chaque nœud.
On répète ce processus en associant à chaque descendant le reste des exemples qui satisfont le test du prédécesseur.

La phase d'apprentissage consiste à la construction de l'arbre de décision avec sa racine, ses nœuds et toutes ses feuilles représentant l'ensemble des règles.
En ce qui concerne la phase de classification : pour classer un nouveau document, il suffit de simplement de parcourir l'arbre selon les réponses aux différents tests pour le document.
L'évaluation de la qualité de l'arbre se fonde le plus souvent sur le taux d'erreur de classification. Ce taux de cas mal classés par les règles, calculé sur les données d'apprentissage ou des données test est évidemment pertinent comme choix des conclusions et validation des règles issues d'arbres (Pisetta & all, 2007).

3.2.5.2- Architecture d'un arbre de décision

Un arbre de décision a pour but d'expliciter une catégorisation dans le cas où les textes classés sont décrits par un ensemble de termes qui représentent les propriétés caractéristiques des textes. Un arbre de décision est un arbre dont les nœuds sont :

- Soit des feuilles contenant des objets appartenant tous à une même catégorie. Les feuilles représentent donc les classes d'une même catégorie C (sous ensembles de C).
- Soit des nœuds de décision qui partitionnent les données suivant plusieurs sous ensembles, chaque sous-ensemble correspondant à un résultat d'une fonction de test, cette fonction caractérisant le nœud de décision. (Mari & Napoli, 1996)

3.2.5.3- Algorithme de construction

L'algorithme d'apprentissage de construction des arbres de décision prend en entrée un ensemble de textes déjà classés sous forme de couples (texte, classe) et présente en sortie un arbre de décision. L'algorithme procède de façon descendante : il démarre du nœud initial (racine), et récursivement, génère les nœuds fils jusqu'aux feuilles.
L'algorithme de construction d'un arbre de décision est basé sur l'hypothèse diviser et conquérir (divide and conquer). Il s'agit de partitionner un corpus C d'apprentissage en plusieurs sous ensembles C_1, C_2,..,C_N, où chaque C_i correspond à un résultat du test utilisé. La procédure est répétée récursivement sur tous les sous-ensembles jusqu'à ce que tous les nœuds terminaux ne contiennent que des objets d'une même classe. (Mari & Napoli, 1996).

D'aprés (Breiman et all, 1984) nous devons définir :

- Un ensemble de questions à poser aux individus (dans notre cas c'est les textes) ; nous choisissons des questions telles que chaque individu peut y répondre par l'affirmative ou la négative ; suivant sa réponse, l'individu est transféré dans le nœud fils correspondant à la réponse « oui » ou dans le nœud fils correspondant à la réponse « non » (Une question Q de la forme : « *les individus contiennent-ils le terme* x *?* » est posée à chaque individu du nœud *S*. Cette question permet de subdiviser le nœud *S* en deux nœuds fils $S_{(OUI)}$ et $S_{(NON)}$ qui comprennent respectivement les individus de *S* qui contiennent et qui ne contiennent pas le terme *x*.
- Une règle pour déterminer les questions à poser aux individus ;
- Un critère d'arrêt déterminant l'ensemble des feuilles de l'arbre.

Le critère de sélection d'une question est souvent fonction du gain en entropie observé avant et après l'affectation des individus dans de nouvelles partitions suivant la question considérée (Quinlan, 1986), (Quinlan, 1993). L'entropie est utilisée comme critère de sélection des questions pour subdiviser les nœuds de l'arbre. Les individus à répartir dans les feuilles de l'arbre sont, ici, des documents.

Ainsi pour induire l'arbre, il faut rechercher à chaque nœud l'éclatement qui produirait le meilleur gain en termes pour la sélection des règles, pour cela deux mesures statistiques sont utilisées : l'entropie et le gain d'information.

3.2.5.4- L'entropie et le gain d'information

L'entropie permet de mesurer l'homogénéité des exemples. Si l'entropie vaut 0, alors tous les exemples appartiennent à la même classe (par exemple Oui). Si l'entropie vaut 1, alors c'est qu'il y a autant d'exemples positifs que d'exemples négatifs.
L'entropie est définie par :

Entropie(S) = -p/N log2 p/N - n/N log2 n/N
Où S est l'ensemble des exemples (**S**amples), de taille **N**,
p (exemples **p**ositifs) est le nombre d'exemples classés Oui,
et n (exemples **n**égatifs) est le nombre d'exemples classés Non dans l'ensemble **S** des **N** exemples.
N.B : on admet 0 log 0 = 0.

Le gain d'information peut être traduit par la quantité d'information apportée à la classe par le terme choisi. Une valeur importante du gain d'information apporté par le terme implique une quantité d'informations importante pour classer les documents avec cet attribut qui est nécessaire.
La mesure appelée gain d'information calcule la réduction attendue de l'entropie des exemples si un attribut particulier est utilisé. L'algorithme de classification calcule cette valeur pour chaque attribut et choisit alors celui qui réduit le plus l'entropie, c'est à dire celui qui permettra le plus nettement possible de séparer les exemples qui restent.

Information Gain(S,A) = Entropie(S) - Somme sur les valeurs de A de (|Sv| * Entropie(Sv) / |S|)
Où S = l'ensemble des exemples,
A est l'attribut utilisé,
Sv = le sous-ensemble de S dont l'attribut a la valeur v.

3.2.5.5- Évaluation des arbres de décision

Les arbres de décisions ont l'avantage d'offrir à la fois un bon pouvoir explicatif (de meilleurs performances de classification), généralement du même niveau des meilleurs techniques de classification et un bon pouvoir descriptif, dans la mesure où on peut interpréter l'arbre comme un ensemble de règles de décision. Pour cela, il suffit de partir de la racine de l'arbre au plus haut et de former tous les chemins possibles pour obtenir une feuille d'une classe.
Nous obtenons donc des règles logiques, sous forme normale disjonctive (SI .. ALORS ..SINON...).

- Si (x=1) et (y=0) alors C_1
- Si (x=1) et (y=1) alors C_2
- Si ((x=0) et (y=1)) ou ((x=0) et (z=1)) alors C_3
  etc...

Ainsi le modèle aura la forme de fonction de décision la plus interprétable par des humains.
D.A. Zighed, et R. Rakotomalala dans (Zighed & Rakotomalala, 2000) évoquent les principales caractéristiques des arbres de décision à savoir : Performance du classifieur du point de vue rapidité et qualité des résultats, tous les types de données sont supportés, capable d'être développés dans tous les environnements de fouille de données et sans oublier bien sûr leur pouvoir descriptif (Lisibilité du résultat)**.**

Cependant, les arbres de décision ont comme principal inconvénient d'être souvent grands, c'est à dire de posséder beaucoup de feuilles puisque on crée un nœud de décision pour chaque résultat possible d'un test. Lorsque l'arbre est très développé, ça devient illisible et l'on perd ainsi en pouvoir explicatif. Il faut alors recourir à des méthodes automatiques d'extraction de règles à partir de l'arbre (Mari & Napoli, 1996).
Ajouté à d'autres inconvénients qui peuvent être soulignés comme leur sensibilité au nombre de classes important qui peut dégrader les résultats, et la non-évolutivité dans le temps, puisque un changement dans les exemples d'entrées exige le relancement de la phase d'apprentissage sur le tout le corpus (anciens exemples et nouveaux exemples). Sans oublier de souligner l'inconvénient causé par le fait que les règles ou arbres de décision ne peuvent prendre que des décisions binaires (Moutarde, 2008).

Les arbres de décision sont des instruments privilégies d'exploration de données, que ce soit en termes de classement ou de description. Dans ce dernier cas, plutôt que de prédire la valeur de la réponse, il s'agit de repérer les profils typiques des individus appartenant à chacune des classes de la variable à prédire. Nous inversons en quelque sorte le problème en cherchant à caractériser les profils propres à la classe, plutôt que la classe à partir du profil (Pisetta & all, 2007). Notons enfin, que le domaine d'utilisation des arbres de décision dépasse le cadre de l'apprentissage supervisé puisque il y a des applications de classification par arbres de décision non supervisés exploités dans le cadre de la recherche documentaire (Bellot, 2000).

3.2.6- Les approches neuronales

3.2.6.1- Présentation de l'approche

Les réseaux de neurones artificiels sont habituellement utilisés pour des tâches de classification. Par analogie avec la biologie, ces unités sont appelées neurones formels.
Un neurone formel est caractérisé par :

- Le type des entrées et des sorties ;
- Une fonction d'entrée ;
- Une fonction de sortie.

Le connexionnisme peut être défini comme le calcul distribué d'unités simples, regroupées en réseau. Un réseau de neurone est un ensemble d'éléments ou unités extrêmement simples (neurones) se comportant comme des fonctions de seuil, suivant une certaine architecture ; Chaque neurone prend en entrée une combinaison des signaux de sortie de plusieurs autres neurones, affectés de coefficients (les poids) ;
L'apprentissage s'effectue sous le contrôle des associations prédéfinies entre documents (entrées du réseau) et classes (sorties du réseau) qui fixent le comportement du réseau souhaité. La différence entre le comportement réel et désiré est une erreur qui sera à la base de l'apprentissage sous la forme d'une fonction de coût ou d'un signal d'erreur. Dans ce cas, l'apprentissage s'effectue en réajustant chaque fois les poids Wi.
Donc les algorithmes d'apprentissage permettent de calculer automatiquement les poids qui correspondent en réalité à des paramètres permettant de définir les frontières des classes.
Une structuration en couches effectue en cascade différents traitements sur un ensemble de données. Ces données sont présentées sur une couche terminale, appelée couche d'entrée; elles sont ensuite traitées par un nombre variable de couches intermédiaires ou couches cachées. Le résultat est exposé sur l'autre couche terminale, la couche de sortie.

3.2.6.2- Le perceptron

Historiquement, le perceptron est le premier réseau efficace qui a été proposé et étudié en détail. Il comprend une couche d'entrée et une couche de sortie. Les connexions entre ces deux couches sont bidirectionnelles et variables et. Les neurones ont des sorties binaires. Les neurones de la couche de sortie réalisent une fonction à seuil. C'est la couche de sortie qui assure l'opération de classification (Leray, 2006).
Les connexions sont ajustées par un apprentissage supervisé selon le principe de correction d'erreurs : si un neurone de la couche de sortie n'est pas dans l'état désiré, toutes les connexions du neurone avec ceux de la couche d'entrée sont augmentées ou diminuées selon le type d'action que le neurone de la couche d'entrée effectuait sur le neurone de sortie en question, dans le but de favoriser une réponse connue au préalable.

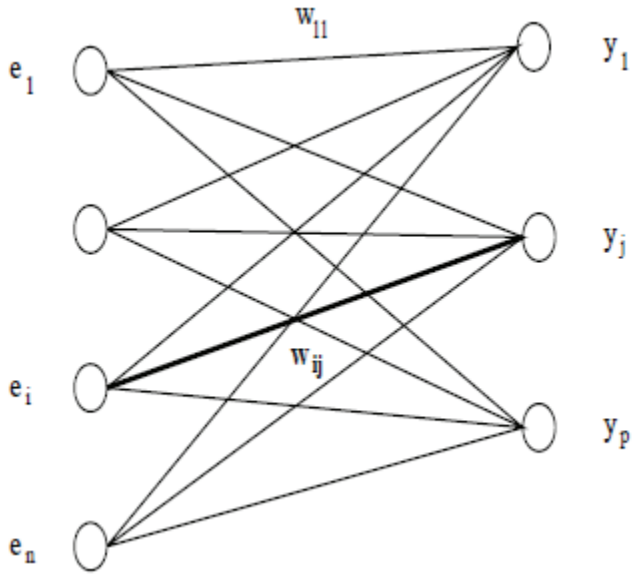

Figure 3.5 : Perceptron monocouche

D'autres réseaux de type perceptron multicouche sont également très utilisés pour les tâches de classification.

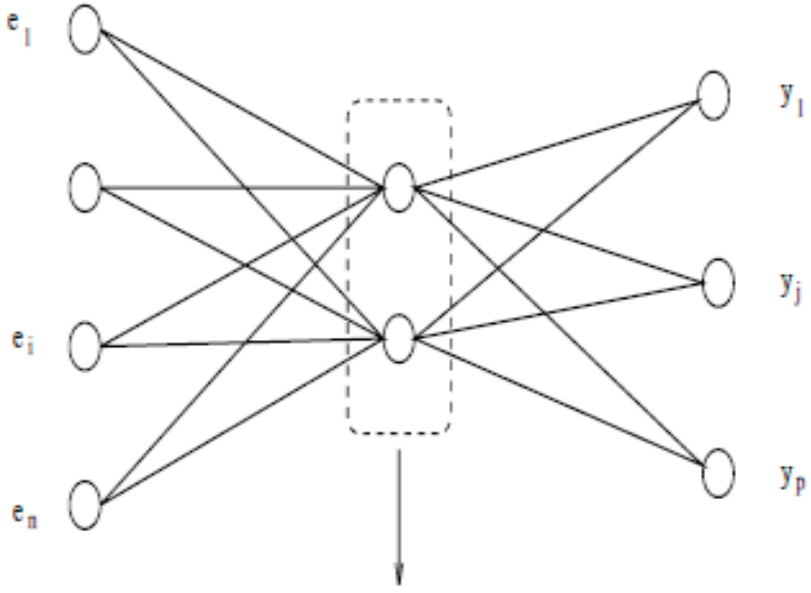

Figure 3.6 : Perceptron multicouche

3.2.6.3- Autres réseaux à couches

Les réseaux à fonction à base radiale (Radial Basis Function ou RBF) sont des réseaux à trois couches. la couche d'entrée qui renvoie les exemples d'entrées sans changement , la couche cachée RBF qui contient les neurones RBF est en ensemble de neurones réalisant une convolution avec une fonction gaussienne, et une troisième couche de sortie qui réalise simplement une combinaison linéaire des neurones de la couche cachée.
Chaque couche est complètement connectée à la suivante.

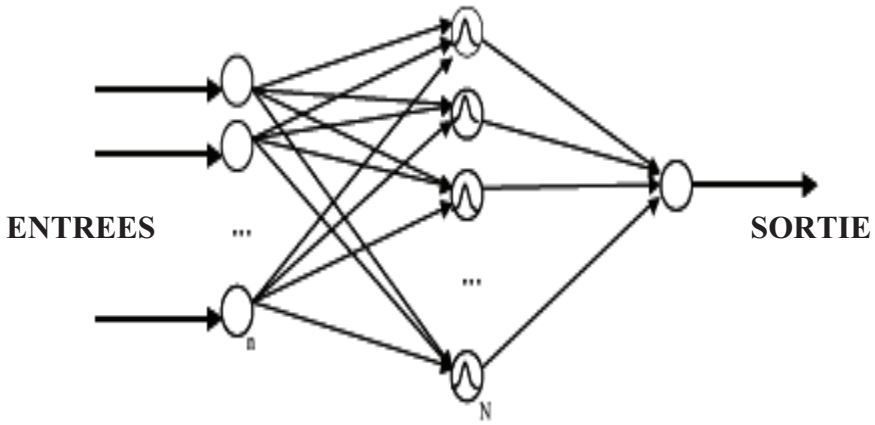

Figure 3.7: Radial Basis Function (RBF)

Les facteurs principaux d'un réseau RBF à ajuster c'est : Le nombre de neurones de la couche cachée RBF, les poids des liaisons des neurones RBF et les neurones de sortie, la position des centres des gaussiennes correspondantes aux neurones RBF, la largeur de ces gaussiennes.

3.2.6.4- Classification à base des réseaux de neurones

Plusieurs travaux, dans le domaine de classification automatique de textes, à base des approches neuronales ont été évoqués.
Une approche fondée sur les réseaux de neurones a été proposée dans la thèse de (Wiener, 1993) dont les résultats ont été repris dans (Wiener & all, 1995). Deux architectures neuronales sont proposées et testées sur le corpus Reuters.
La première architecture est un perceptron multi-couche avec une couche de neurones cachés et un neurone de sortie; un réseau de neurones différent est construit pour chaque classe.
Pour la deuxième architecture, les classes du corpus Reuters sont regroupées en cinq grands ensembles (*agriculture, energy, foreign exchange, government, metals*). Un réseau est ensuite utilisé pour déterminer à quel ensemble appartient un document, puis cinq réseaux différents sont construits pour déterminer, à l'intérieur d'un ensemble, la classe exacte du texte. Cette architecture a l'avantage de permettre à chacun des cinq réseaux d'être spécialisé et d'utiliser une représentation particulièrement adaptée pour distinguer des classes proches.
Cette deuxième architecture améliore les résultats, mais elle nécessite un découpage manuel des classes pour déterminer les ensembles et n'est réalisable que sur un corpus pour lequel le nombre de classes est connu à l'avance et n'évolue pas.
Dans l'ensemble de cette étude, le surajustement est limité en considérant un terme de pénalisation dans la fonction de coût conjointement avec la méthode de l'arrêt prématuré ;

Cette étude montre notamment que, même lorsque le nombre de neurones cachés est nul, le modèle peut être surajusté. La mise en œuvre d'une procédure d'arrêt prématuré limite ce surajustement et améliore significativement les résultats.

3.2.6.5- Critiques de l'approche

A partir des expériences à bases des réseaux de neurones, il est possible de tirer plusieurs conclusions :

- D'après les études précédentes, l'ajout de neurones cachés n'améliore pas vraiment les performances du classifieur.
- Il est nécessaire de se protéger du surajustement, même pour le modèle sans neurone caché, par une méthode de régularisation qui peut prendre la forme d'un terme de pénalisation dans la fonction de coût ou d'une procédure d'arrêt prématuré.
- La combinatoire des combinaisons linéaires et des fonctions de seuil appliquées récursivement donne un classificateur complexe.
- Et enfin une caractéristique des réseaux connexionnistes qui représente leur inconvénient principal c'est leur très faible pouvoir descriptif puisque un réseau de neurone est une sorte de boîte noire non interprétable par ses utilisateurs.

3.2.7- Naïve Bayes

L'algorithme Naïve Bayes (NB), est une autre méthode bien utilisée en apprentissage, elle est également employée dans la classification automatique de textes. NB est très connu pour son efficacité, sa facilité d'implémentation et ses résultats considérables. Pour toutes ces raisons et bien d'autres avantages, nous l'avons adopté dans nos travaux de recherches.
Une description détaillée sera accordée ultérieurement dans la chapitre 6 (section 6.3.2).

3.2.7.1- Description de l'approche

L'algorithme Naïve Bayes (NB) est une méthode basée sur le modèle probabiliste, il vise à estimer la probabilité conditionnelle d'une catégorie sachant un document et affecte au document la (ou les) catégorie(s) la (les) plus probable(s). Ainsi ce classifieur va donc tenter d'estimer la probabilité qu'un document d fasse partie de la classe c.
Un modèle probabiliste de classification est un modèle qui permet le calcul pour chaque classe $c \in C$ et chaque document $d \in D$ la probabilité $\mathbb{P}(c / d)$ que le document soit assigné à cette classe.
Le nom Naïve Bayes découle du fait que l'algorithme utilise le théorème de Bayes sans toutefois prendre en compte les dépendances existantes entre les variables (Dans notre cas les termes c'est des mots ou n-grammes, etc..); de ce fait, ses suppositions sont dites naïves. Donc la partie naïve de ce modèle est l'hypothèse d'indépendance des termes, c'est à dire que la probabilité conditionnelle d'un terme sachant une catégorie est supposée indépendante de cette probabilité pour les autres termes.
Cette hypothèse fait que la catégorisation par NB est plus efficace que la complexité exponentielle des approches bayesiennes non naïves qui utilisent des combinaisons de mots comme prédicteurs, (McCallum & all, 1998) (Yang & Liu, 1999), (Hertzmann, 2004), (Chethan & all, 2007).

Dans un contexte probabiliste bayésien général, le choix de la classe d'une nouvelle instance ***A*** est réalisé par la règle du maximum a posteriori (MAP). L'estimation des probabilités est réalisée en général par maximisation de la vraisemblance conditionnelle de l'ensemble d'apprentissage par la formule $\mathbb{P}(\boldsymbol{B})*\mathbb{P}(\boldsymbol{A/B})$ qui fait apparaître deux termes : la vraisemblance

de l'observation ***A*** conditionnellement à ***B*** et la probabilité a priori de ***B***. Dans un cadre supervisé, ces deux distributions sont estimées à partir des exemples d'apprentissage.
Si l'on considère un problème à deux classes c_1 et c_2, L'apprentissage consiste à déterminer la distribution de probabilité connaissant les données d'apprentissage : une probabilité fixée a priori, et, une fois que les données d'apprentissage ont été observées, cette probabilité a priori est transformée en probabilité a posteriori grâce au théorème de Bayes.
Le théorème de Bayes permet donc de calculer les probabilités a posteriori connaissant les distributions des observations a priori.
Notons que la probabilité a posteriori est la probabilité conditionnelle $P(c_j/d_i)$ de la classe ci connaissant d_i. En d'autres termes, quand on a un exemple d'entrée d_i (dans ce cas d_i est un texte), $P(c_j/d_i)$ représente la probabilité que d_i appartienne à la classe c_j.
Pour la probabilité a priori c'est la probabilité conditionnelle $P(d_i/c_j)$. Cela représente la probabilité d'avoir d_i comme entrée quand on sait que l'on est dans la classe c_j.
La classification d'un exemple s'obtient alors 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 à 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.
Fondés sur l'idée qu'on peut estimer la probabilité qu'un document appartient à une classe en connaissant la probabilité qu'une classe correspond à ce document.
La formule de bayes permet « d'inverser » la probabilité conditionnelle :

$$P(c_j|d_i) = \frac{P(c_j) \times P(d_i|c_j)}{P(d_i)}$$

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 (Dans ce cas pourcentage de documents) appartenant à la classe c_j dans le corpus d'apprentissage.

$$P(c_j) = \frac{N(c_j)}{N}$$

Quant à la classification, l'estimation $P(d_i / c_j)$ est calculée à partir des deux formules suivant le modèle utilisé en remplaçant les probabilités par leur estimateur et **la classe c_j ayant la probabilité la plus élevée est choisie.**

3.2.7.2- Critiques de l'approche

L'algorithme NB est connu par son efficacité et sa simplicité qui revient à l'effet admis, d'indépendance entre les différents descripteurs et à cause de cette hypothèse d'indépendance des mots dans ce modèle, on le qualifie souvent de "Naïve", "Idiot", "Simple". En général, ce type d'algorithmes permet de faire le même travail de classification que les autres algorithmes qui ont déjà prouvés dans le domaine. Ce classifieur est très favorable pour les documents courts qui donne des résultats très intéressants, néanmoins ces performances sont réduites quand il s'agit d'un vocabulaire important à traiter, ainsi le manque d'une meilleure prise en compte de la taille des documents, fait que ses performances en qualité de classement se dégradent avec l'augmentation du nombre de caractéristiques. En effet, si le nombre de termes

augmente, alors le nombre des dépendances entre l'ensemble des termes augmentent aussi, et donc, la vérification de l'hypothèse de Naïve Bayes diminue. (Hilali, 2009)

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).
Notons aussi que ce modèle ne prend pas en compte l'ordre des mots dans la modélisation des documents. Intuitivement, cela apparaît comme une hypothèse très forte et peu réaliste. Si pour la tâche de classification, cette hypothèse peut paraître représenter un compromis entre la puissance du modèle et sa simplicité, elle rend impossible différentes tâches comme le résumé automatique, l'extraction de passage, etc. Dans les années 80, (Jelinek & Mercer, 1980) ont proposé d'abandonner cette hypothèse d'indépendance (ou de dépendance à l'ordre 0) afin de prendre en compte localement l'ordre des mots. Les modèles développés alors reposent sur une hypothèse d'indépendance à un ordre supérieur. Cette hypothèse s'écrit :
$P(t_i/t_{i-1},....,t_1, c_j) = P(t_i/t_{i-1},....,t_{i-n}, c_j)$ ou n représente l'ordre de la dépendance.
Dans le cas des données textuelles, les probabilités $P(t_i/t_{i-1},....,t_{i-n}, c_j)$ correspondent à la probabilité de voir dans un document le mot t_i précédé de la séquence de mots $t_{i-1},....,t_{i-n.}$

Notons enfin, qu'un certain nombre de modifications à l'algorithme ont été proposées dans le but d'élargir son utilisation et améliorez ses performances.

3.2.8- Les méthodes mixtes et Boosting

3.2.8.1- Présentation de l'approche

Plusieurs auteurs dans leurs travaux récemment réalisés, ont proposé des modèles hybrides pour la classification de données textuelles pour combiner les résultats de classifieurs simples (et pas très bons) pour donner des classifieurs plus complexes (et bien meilleurs) :

- Procédures de vote : Plusieurs classifieurs s'entrainent sur le même corpus pour tenter de classer chacun de sa manière un même nouveau document, la classe qui a été choisit par le plus grand nombre de classifieurs va être attribuée à ce document.
- Boosting : Est une instance des classifieurs par comités dont le principe est d'associer les résultats de plusieurs classifieurs pour obtenir un résultat plus intéressant. Les différents classifieurs peuvent correspondre à différents algorithmes ou au même algorithme utilisé avec différents sous-échantillons du corpus d'apprentissage. Dans cette approche, les différents classifieurs sont appris séquentiellement, à chaque étape, le corpus d'origine est échantillonné suivant une distribution qui favorise le tirage des documents d'entrainement mal classés par le classifieur construit à l'étape précédente. Les classifieurs se focalisent ainsi de plus en plus sur les exemples difficiles à catégoriser. Lors de la phase de test, tous les classifieurs sont utilisés et un vote pondéré par les taux de performance de chaque système est effectué, comme dans les procédures de vote. 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.

3.2.8.2- Evaluation de l'approche

Les expériences menées montrent que, en pratique, les performances en généralisation sont très bonnes et les risques d'avoir du sur-apprentissage est très minime. En revanche le problème d'efficacité des résultats en terme de temps peut être posé dans le cas où on opte

pour un nombre important de classifieurs pour améliorer les résultats. Comme SVM, NB, kPPV, le boosting a été adapté à la catégorisation de textes avec succès.

3.2.9- Autres méthodes

Il y a d'autres approches plus ou moins utilisées dans le cadre de classification de textes mais elles sont surtout orientées vers la recherche d'information, résumé automatique, etc...

- Comme par exemple la régression logistique :

L'entrée du classifieur logistique est une représentation des phrases en un vecteur de scores, Les paramètres sont appris en maximisant la log-vraisemblance binomiale des documents d'apprentissage. La régression logistique a déjà été utilisée avec succès en résumé automatique (Usunier & all, 2005) et a montré de bonnes performances en tant qu'algorithme de combinaison de caractéristiques.

- Ou les Modèles de Markov Cachés (MMC) :

Lorsque le nombre de classes est connu, ce dernier problème peut également être résolu en utilisant des Modèles de Markov cachés. Un modèle de Markov caché est un modèle probabiliste dont le but est de modéliser un processus séquentiel. Nous considérons ici uniquement le cas des MMCs dans des espaces discrets, ce qui correspond à l'utilisation la plus courante de ces modèles avec des données textuelles.

Pour construire des regroupements en k classes, on suppose de nouveau que chaque séquence est issue d'un parcours dans un graphe à k états (un par classe), chaque état étant caractérisé par un modèle de génération de x qui lui est propre.

On définit un MMC comme un triplet $\{A, B, \Pi\}$: L'ensemble des états possibles du processus, l'ensemble des observations possibles, une matrice de probabilités initiales.

Il est habituel de représenter les MMC sous forme graphique.

Les MMCs ont été utilisés pour différentes problématiques de la RI et du traitement du langage naturel. (Miller & all, 1999) pour la recherche documentaire et (Amini, 2001) qui a développé un modèle d'extraction d'information (professions, personnes, etc..) dans des documents textes.

- Ou encore les Réseaux bayésiens :

Les réseaux bayésiens se présentent comme un formalisme de représentation graphique des dépendances conditionnelles entre des variables aléatoires. Historiquement, ils ont été développés dans le cadre de la prise en compte de l'incertain pour la prise de décisions. Ils représentent une extension du classifieur Naïve Bayes.

Un réseau bayésien est un graphe orienté sans cycles $G = (X, U)$ où X est l'ensemble des sommets et U l'ensemble des arcs qui définissent la fonction pa(Xi). La fonction pa(Xi) renvoie pour toute variable X_i l'ensemble de ses parents.

La phase dite d'apprentissage consiste à l'estimation des différentes probabilités conditionnelles à priori à partir d'un corpus d'apprentissage. L'apprentissage de ces paramètres est souvent complexe et coûteux en terme de calculs.

La phase dite d'inférence correspond au calcul de la probabilité d'une observation quelconque. Cette inférence peut-être exacte ou estimée.

3.3- Mesures de similarité et formules pour calcul de distance

Plusieurs méthodes de classification présentées précédemment et particulièrement les méthodes géométriques s'appuient sur le principe des mesures de distance.

Les bons résultats de ces méthodes sont démontrés dans plusieurs travaux, en revanche la faiblesse des méthodes utilisant les mesures de distance apparaît dans le cas des espaces de travail importants (grand nombre de documents, par exemple dans le cas du web). Les

performances de ces techniques diminuent considérablement non pas en qualité des résultats mais en rapidité des calculs.
Dans ce contexte, il existe plusieurs variantes de distance, et divers mesures de similarités entre documents ou classes qui peuvent être utilisés, dont l'influence sur les performances d'un système de catégorisation est démontré.
Dans ce qui suit, on va évoquer les différents choix possibles pour la distance, pour ensuite présenter les mesures de similarités les plus utilisés dans les domaines de la recherche d'information et la classification.

3.3.1- Calcul de distance

3.3.1.1- Définition de la distance

Une distance est une fonction de $E \times E$, où E est un espace vectoriel.

Cette fonction est caractérisée par les propriétés suivantes :

$D(x, y) \geq 0$ $\qquad$ $D(x, y) = 0 \Longleftrightarrow x = y$

$D(x, y) = D(x, y)$ $\qquad$ $D(x, y) \leq D(x, z) + D(z, y)$

x, y, z sont des éléments de l'espace E. (Saporta, 1990)

Dans le contexte de catégorisation, ces éléments sont soit des textes soit des classes.

3.3.1.2- Variantes de distance

Une formule générale est connue pour mesurer les distances dans les espaces vectoriels c'est la grandeur de Minkowski :

$$D_k(x, y) = \sqrt{(\sum_i | xi - yi |^k)^{1/k}}$$

A partir de cette formule très générale, plusieurs distances connues en pratique sont déduites :

- Comme la distance euclidienne, dans le cas où k = 2, exprimée par :

$$D_e(x, y) = \sqrt{\sum_i (xi - yi)^2}$$

- Où la distance de Manhattan, dans le cas où k = 1, formulée par :

$$D_m(x, y) = |x_1 - y_1| + |x_2 - y_2| + ... + |x_n - y_n|$$

- Où aussi la distance de Chebyshev, dans le cas où $k = \infty$, définie par :

$$D_c(x, y) = \max\{ |x_1 - y_1| , |x_2 - y_2| , ... , |x_n - y_n| \}$$

Les résultats fournis par un classifieur peuvent varier suivant l'utilisation de telle ou telle distance, comme par exemple dans les kPPV, Radwan JALAM dans (Jalam, 2003) confirme bien que le plus proche voisin peut varier selon la distance utilisée, ainsi la distance

euclidienne favorise bien les voisins dont tous les descripteurs sont assez proches, cependant la distance de Manhattan permet de tolérer une distance importante sur l'un des descripteurs.

3.3.2- Mesures de similarité

La problématique de classification automatique de textes peut se résumer en une formalisation de la notion de similarité textuelle, soit en d'autres termes : construire un modèle mathématique capable de représenter, pour ensuite comparer, la sémantique des textes. Si cette notion de similarité sémantique est un processus souvent intuitif pour l'homme, elle résulte d'un processus complexe et encore mal compris du cerveau. Le but de la recherche sur la catégorisation est donc de trouver un algorithme permettant d'attribuer les nouveaux textes à une classe avec le plus petit taux d'erreur possible, sans toutefois associer un texte à trop de classes. Dans un tel contexte, une mesure de similarité textuelle permet d'identifier la ou les catégories les plus proches du texte à classer. S'il existe plusieurs techniques pour la représentation des textes, il en est de même pour les formules adoptées comme élément de mesure. Quatre d'entre elles se sont illustrées dans le domaine : Nombre de mots communs entre un document et une classe, Okapi, Cosinus et Kullback&Liebler. La première mesure est basique qui n'est qu'une simple comparaison entre les profils des objets (texte ou classe) pour extraire les intersections entre ses objets (Nombre de mots en commun). La seconde est une variante de la mesure Okapi testée avec succès par (Wilkinson & all, 1996) (Une description peut être trouvée dans (Bellot, 2000). Les deux autres mesures à savoir Cosinus et Kullback&Liebler sont présentées dans ce qui suit.

3.3.2.1- Cosinus

Dans les modèles vectorielles, la mesure du cosinus est la plus utilisée pour définir la similarité entre un texte et une classe (Sebastiani, 2002) en raison de la stabilité de ses résultats sur des corpus variés. La similarité de deux éléments (dans notre cas documents) qu'on veut comparer, peut alors être définie par le cosinus de l'angle séparant les vecteurs des deux éléments (Salton & McGill, 1983). Par rapport à un simple produit scalaire, cette mesure présente l'avantage de normaliser les scores de chaque objet en fonction de sa taille, elle-même pondérée par le poids des termes. Le résultat renvoyé est facilement exploitable ensuite car c'est une valeur située entre 0 et 1. La valeur 1 indiquant une similarité maximum (les deux objets sont identiques) et 0 une similarité nulle (les deux objets n'ont absolument rien en commun). Cette mesure est égale au produit scalaire divisé par les deux vecteurs sont qui sont déjà normalisés.

$$\mathrm{Cosinus}(i,j) = \frac{\sum_{w \in i \cap j} \mathrm{TFIDF}_{w,i} \times \mathrm{TFIDF}_{w,j}}{\sqrt{\left(\sum_{w \in i} \mathrm{TFIDF}_{w,i}^2\right)} \times \sqrt{\sum_{w \in j} \mathrm{TFIDF}_{w,j}^2}}$$

Avec : ***w*** un terme, ***i*** et ***j :*** les deux objets (profils documents ou classes) à comparer. $TFIDF_{w,i}$ le poids du terme ***w*** dans ***i*** et $TFIDF_{w,j}$ le poids du terme ***w*** dans ***j***.

Ce qui peut ce traduire de la façon suivante :
« ***Plus on a de termes communs et plus ces termes communs ont des pondérations fortes, plus la similarité sera proche de 1, donc forte et vice versa. »***

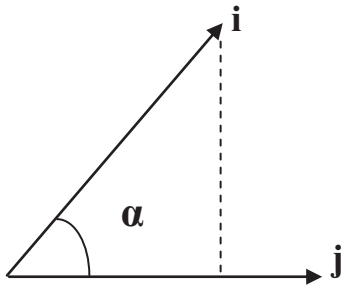

Figure 3.8 : La mesure de similarité Cosinus
i et j sont respectivement les vecteurs représentant le texte et la classe

3.3.2.2- Kullback&Liebler (la mesure d'entropie relative)

Kullback et Liebler ont étudié en 1951 une mesure statistique d'information appelée fonction de discrimination en prenant en considération deux distributions de probabilité.
La mesure Kullback&Liebler connu aussi sous le nom de l'entropie relative est une mesure qui calcule la divergence entre deux distributions de probabilité. En effet la divergence entre deux probabilités P et Q sur un ensemble fini X est définie comme suit :

$$D(P\|Q) = \sum_{x \in X} P(x) \times \log \frac{P(x)}{Q(x)}$$

Il faut noter que cette divergence n'est pas symétrique ($D(P\|Q) \neq D(Q\|P)$).
Donc la divergence symétrique Kullback&Liebler est définie comme suit :

$$D(P\|Q) = \sum_{x \in X} \left(\left(P(x) - Q(x)\right) \times \log \frac{P(x)}{Q(x)} \right)$$

Tout de même une version symétrisée de la distance de Kullback-leibler pour faciliter la comparaison avec les autres mesures de similarité existe dans (Haddad, 2002)
Cette mesure de similarité a été utilisée dans différents domaines tel que le traitement des langages naturels (Carpinto & all, 2001), la recherche d'information pour l'identification des thèmes (Bigi & all, 2000) ainsi que la reconnaissance de la parole (Dagan, 1999) et (Dagan & all, 2005).
Dans le contexte de catégorisation de textes, cette mesure est utilisée pour calculer la distance entre le profil du texte et le profil de la classe comme suit :

$$KLD(c_i, d_j) = \sum \left\{ \left(P(t_k, c_i) - P(t_k, d_j)\right) \times \log \left(\frac{P(t_k, c_i)}{P(t_k, d_j)} \right) \right\}$$

Dans son calcul, quatre cas sont pris en considération :

- ($t_k \in d_j$) et ($t_k \in c_i$) i.e : le terme t_k apparaît dans le profil de la catégorie et dans le profil du document.
- ($t_k \in d_j$) et ($t_k \notin c_i$) i.e : le terme t_k apparaît dans le profil du document mais n'apparaît pas dans le profil de la catégorie.
- ($t_k \notin d_j$) et ($t_k \in c_i$) i.e : le terme t_k n'apparaît pas dans le profil du document mais apparaît dans le profil de la catégorie.

- ($t_k \notin d_j$) et ($t_k \notin c_i$) i.e : le terme t_k n'apparaît pas dans le profil du document et n'apparaît pas non plus dans le profil de la catégorie.

La probabilité d'apparition d'un terme t_k dans un profil de catégorie est définie comme suit :

$$\begin{cases} P(t_k, c_i) = \dfrac{tf(t_k, c_i)}{\sum_{x \in c_i} tf(t_x, c_i)} & \textbf{\textit{Si le terme } t_k \textit{ apparaît dans le profil de } c_i.} \\ \varepsilon \text{ (epsilon)} & \textbf{\textit{Sinon.}} \end{cases}$$

De même, La probabilité d'apparition d'un terme t_k dans le profil du document est définie comme suit :

$$\begin{cases} P(t_k, d_j) = \dfrac{tf(t_k, d_j)}{\sum_{x \in c_i} tf(t_x, d_j)} & \textbf{\textit{Si le terme } t_k \textit{ apparaît dans le profil de } d_j.} \\ \varepsilon \text{ (epsilon)} & \textbf{\textit{Sinon.}} \end{cases}$$

Où :

- $P(t_k, c_i)$ est la probabilité conditionnelle d'un terme dans une catégorie.
- ε est une probabilité accordée aux termes qui n'apparaissent ni dans le document, ni dans la catégorie ;

Pour chaque catégorie, il est nécessaire de normaliser la distance, parce que les catégories sont de tailles différentes. Par conséquent, nous utiliserons la distance Kullback&Liebler normalisée :

$$KLD^*(c_i, d_j) = \frac{KLD(c_i, d_j)}{KLD(c_i, 0)}$$

Où : $KLD(c_i, 0)$ représente la distance entre la catégorie et un document vide.
Finalement, après avoir calculer la distance $KLD^*(c_i, d_j)$ entre le document à catégoriser et toutes les catégories, le document sera assigné à la catégorie la plus proche :

$$H_{KLD^*}(d_j) = \arg\min_{c_{i \in C}} KLD^*(c_i, d_j)$$

3.3.2.3- Synthèse sur les mesures de similarité

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, plusieurs études ont constaté la nette supériorité de cosinus par rapport aux autres mesures dans la majorité des cas. Dans (Jones & Furnas, 1987) une analyse géométrique des différentes mesures de similarité existantes est effectuée et elle conclue que le cosinus semble être le choix le plus judicieux car le moins sensible aux cas particuliers qui génèrent des comportements illogiques avec les autres mesures. Néamoins un dysfonctionnement de cosinus dans certains cas a déjà été constaté par (Wilkinson & all, 1996), qui proposent de combiner différentes mesures pour améliorer les résultats. En effet Wilkinson propose de combiner linéairement les mesures Cosinus, Okapi et NbMotsCommuns mais en obtenant seulement de légères améliorations. Comme nous pouvons citer aussi un autre élément de mesure appelé Coefficient de Cohérence recommandé par Salton, ce dernier a proposé un procédé d'extraction limité aux expressions de deux mots et se base sur la cooccurrence des termes utilisant un coefficient de cohérence qui représente la proportion des cas de cooccurrence de deux termes.

3.4- Conclusion

La phase de catégorisation de textes et le choix de technique d'apprentissage (Chapitre 3) c'est le cœur du processus de classification automatique de textes, elle est située entre une phase primordiale, de préparation des documents et catégories à l'informatisation (codage des documents), pour le bon fonctionnement du processus (Chapitre2), et une autre phase d'évaluation du ou des classifieurs utilisés aussi importante pour l'amélioration des performances du système (Chapitre 4).
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, qui vont être évoqués dans le chapitre suivant, mais habituellement plusieurs paramètres peuvent influencer le choix de la technique de classification qui est relatif aux attentes des utilisateurs du système. Si l'efficacité est une priorité on peut privilégier un système rapide simple avec des résultats moins performants, et si l'exactitude des résultats et minimiser les erreurs est plus important que l'aspect temps de classification on peut pencher vers d'autres méthodes plus complexes et plus performantes, par contre si on veut avoir un système avec un pouvoir descriptif important qui peut être interprété intuitivement par son utilisateur, alors on va favoriser des modèles compréhensibles tels que les arbres de décision.
Finalement on peut dire qu'il ya des classifieurs plus performants que d'autres mais ce qui est sûr qu'il n'y a pas de classifieur parfait.