

# THESE de DOCTORAT de L'UNIVERSITÉ PARIS 6

**Spécialité :**

**Informatique**

Présentée par

M. Massih-Réza AMINI

pour obtenir le grade de DOCTEUR de l'UNIVERSITÉ PARIS 6

---

Sujet :

**Apprentissage Automatique et Recherche de l'Information :  
application à l'Extraction d'Information de surface et au  
Résumé de texte**

---

soutenue le 13 juillet 2001

devant le jury composé de :

M. Stéphane	CANU	rapporteur
M. Christian	FLUHR	examineur
M. Patrick	GALLINARI	directeur de thèse
M. Christian	JACQUEMIN	rapporteur
M. Yves	KODRATOFF	examineur
M. Djamel	ZIGHED	examineur



## Remerciements

J'adresse mes sincères remerciements à Monsieur Stéphane Canu, professeur à l'INSA de Rouen et à Monsieur Christian Jacquemin, professeur à l'université de Paris XI qui ont accepté d'évaluer ce travail.

Je tiens également à remercier Monsieur Christian Fluhr, professeur à l'INSTN, Monsieur Yves Kodratoff, directeur de recherche à L.R.I et Monsieur Djamel Zighed, professeur à l'université de Lyon II, pour avoir bien voulu participer à mon jury.

Je suis très redevable à mon directeur de thèse Monsieur Patrick Gallinari, professeur à l'université de Paris VI. Je le remercie pour ses précieux conseils et pour son soutien pendant toutes ces années. Il a été un véritable plaisir de travailler avec lui. Sa bonne humeur et sa gentillesse ont également contribué au bon déroulement de mes recherches au sein de son équipe. Je n'oublierai jamais le 25 mars 1996, le jour où il a accepté que je lui présente mes travaux, ce jour était une plaque tournante de ma vie et le point de départ de cette thèse.

L'ambiance détendue qui règne au sein de la « *dream team* » m'a permis de travailler dans de très bonnes conditions. Je souhaite exprimer ma sympathie à tous les membres de l'équipe : ceux qui m'ont précédé, Philippe, Yann et tout particulièrement Thierry et Hugo en qui j'ai trouvé plus que des collègues mais de vrais bons amis et ceux qui m'ont succédé. Haifeng, Benjamin, Ihseine, Alda, Hermine, Sanparith, Nadji, Ludovic et Henri.

Enfin merci à ma famille qui m'a épaulée pendant toutes ces années de thèse, tout particulièrement à Fabienne pour ces conseils, à ma chère Hengameh et à ma *fun club* : Maryam et Mona.

**Titre :** Apprentissage Automatique et Recherche d'Information : application à l'Extraction d'Information de surface et au Résumé de Texte.

**Résumé :** la thèse porte sur l'utilisation de méthodes issues de l'apprentissage automatique pour des tâches de recherche d'information dans les textes. Notre motivation a été d'explorer le potentiel des techniques d'apprentissage pour répondre aux demandes d'accès à l'information textuelle liées au développement de grandes bases de données texte et au *web*. Dans ce contexte il est devenu important d'être capable de traiter de grandes quantités de données, d'apporter des solutions diversifiées aux nouvelles demandes des utilisateurs, et d'automatiser les outils qui permettent d'exploiter l'information textuelle. Nous avons pour cela exploré deux directions. La première est le développement de modèles permettant de prendre en compte l'information séquentielle présente dans les textes afin d'exploiter une information plus riche que la représentation sac de mots traditionnellement utilisée par les systèmes de recherche d'information. Pour cela nous proposons des modèles statistiques basés sur des modèles de Markov cachés et des réseaux de neurones. Nous montrons comment ces systèmes permettent d'étendre les capacités des modèles probabilistes classiques de la recherche d'information et comment ils peuvent être utilisés en particulier pour des tâches d'extraction d'information de surface. La deuxième direction explorée concerne l'apprentissage semi-supervisé. Il s'agit d'utiliser pour des tâches d'accès à l'information une petite quantité de données étiquetées conjointement à une masse importante de données non étiquetées. Cela correspond à une situation de plus en plus fréquente en recherche d'information. Nous proposons et analysons des algorithmes originaux basés sur un formalisme discriminant. Nous avons utilisé ces techniques pour le résumé de texte vu sous l'angle de l'extraction de phrases pertinentes d'un document. Ces travaux se sont concrétisés par le développement du Système d'Aide au Résumé Automatique (S.A.R.A.).

**Mots clés :** Accès à l'information textuelle, recherche et extraction d'information, apprentissage, modèles de séquences, apprentissage semi-supervisé, résumé de texte.

**Title:** Automatic Learning and Information Retrieval: Application to surface Information Extraction and Text Summarization.

**Abstract:** The purpose of this work is the application of discriminant learning models to Information Retrieval tasks. Our concern was to explore the potential of learning techniques to handle textual information access needs related to the development of huge databases and Internet. In this context it becomes important to treat huge quantities of data, to provide varied solutions to new user inquiries and to automate tools which allow exploiting textual information. For this, we have explored two directions. The first is the development of models allowing to take into account sequential information present in documents in order to be able to take advantage of an information richer than the usual *bag of words* representation used by classical information retrieval systems. For this, we propose statistical models based on Hidden Markov Models and Neural Networks. We show how these systems allow to extend the capabilities of classical information retrieval probabilistic models and in particular, how they can be used for the surface information extraction tasks. The second direction explored concerns the semi-supervised learning. It is a matter of using a small-labeled data together with a huge unlabeled data to learn systems for information access tasks. This situation is frequently met in information retrieval. We propose and analyze original algorithms based on discriminant formalism. We have used these techniques for the text summarization task regarded as the extraction of the most relevant sentences of a document. This study is concretized by the development of an automatic summarizer system (S.A.R.A.).

**Keywords:** Textual information access, Information retrieval and extraction, Machine Learning, Sequence models, semi-supervised learning, Text summarization.

# Table des matières

<b>Introduction</b>	<b>1</b>
1. Apprentissage et information textuelle.....	1
2. Questions abordées.....	2
3. Modèles de séquences.....	3
4. Apprentissage semi-supervisé.....	3
5. Plan de la thèse.....	5

<b>Notation</b>	<b>7</b>
-----------------	----------

## Partie I : Modèles dynamiques et apprentissage automatique

<b>Chapitre 1. Modèles de l'apprentissage numérique</b>	<b>11</b>
1.1 Introduction.....	11
1.2 Réseaux de neurones.....	12
1.2.1 Introduction.....	12
1.2.2 Unités neuronales.....	13
1.2.2.1 Unités à seuil.....	13
1.2.2.2 Unités linéaires et sigmoïdes.....	14
1.2.3 Algorithmes d'apprentissage.....	15
1.2.3.1 Règle de Hebb.....	15
1.2.3.2 Règle de Widrow-Hoff.....	15
1.2.4 Quelques réseaux.....	16
1.2.4.1 Perceptron et Adaline.....	16
1.2.4.2 Perceptrons multi-couches.....	18
1.3 Modèles de Markov Cachés.....	21
1.3.1 Introduction.....	21
1.3.2 Probabilité d'une séquence d'observation.....	23
1.3.3 Détermination de la séquence d'états optimale.....	24
1.3.4 Apprentissage des paramètres.....	24
1.4 Machines à Vecteurs Support.....	26
1.4.1 Introduction.....	26
1.4.2 MVS Linéaires.....	27
1.4.2.1 Cas séparable.....	27
1.4.2.2 Cas non-séparable.....	29
1.4.3 MVS Non linéaires.....	30
1.5 Conclusion.....	32

1.6 Bibliographie .....	33
<b>Chapitre 2. Apprentissage supervisé et non-supervisé</b>	<b>37</b>
2.1 Introduction .....	37
2.2 Apprentissage supervisé .....	38
2.2.1 Introduction .....	38
2.2.2 Problème d'optimisation .....	38
2.2.2.1 Apprentissage hors-ligne et en-ligne .....	38
2.2.2.2 Procédure stochastique pour l'adaptation et l'apprentissage .....	39
2.2.3 Algorithmes d'optimisation.....	39
2.2.3.1 Critère global.....	39
2.2.3.2 Descente de gradient .....	39
2.2.3.3 Problèmes de régularité .....	40
2.2.4 Etude de la convergence.....	41
2.2.5 Dilemme biais-variance.....	41
2.2.6 Le vote majoritaire .....	45
2.2.6.1 Bagging .....	45
2.2.6.2 Boosting .....	46
2.2.6.3 Un exemple : classification de décharges partielles .....	50
2.3 Apprentissage Non-Supervisé .....	51
2.3.1 Introduction .....	51
2.3.2 Mélange de densités .....	52
2.3.3 Algorithme EM .....	53
2.3.3.1 L'algorithme.....	53
2.3.3.2 Un exemple : Estimation des paramètres d'un mélange gaussien .....	56
2.3.4 Maximum de vraisemblance de classification.....	57
2.3.5 Estimation de paramètres dans le cas de supervision imparfaite .....	58
2.4 Conclusion .....	60
2.5 Bibliographie .....	62
<b>Chapitre 3. Apprentissage avec des exemples étiquetés et non étiquetés</b>	<b>65</b>
3.1 Introduction .....	65
3.2 Apprentissage semi-supervisé par la discrimination .....	66
3.3 Algorithmes non-supervisés et semi-supervisés avec des modèles discriminants .....	67
3.3.1 Décision Dirigée.....	68
3.3.2 Auto Supervision.....	70
3.3.4 Co-Boosting .....	72
3.3.5 Co-Training.....	75
3.4 Algorithmes semi-supervisé avec un modèle génératif.....	76
3.4.1 Maximiser la vraisemblance d'un mélange.....	77
3.4.1.1 Le classifieur Naïve Bayes.....	77
3.4.1.2 L'algorithme semi-supervisé basé sur EM.....	78
3.5 Conclusion .....	79
3.6 Bibliographie .....	81

## Partie II : Fouille de données textuelles

<b>Chapitre 4. Accès à l'information textuelle</b>	<b>85</b>
4.1 Introduction .....	85
4.2 Extraction de l'Information .....	86
4.2.1 Les techniques de bases .....	87
4.2.1.1 Analyse locale du texte .....	89
4.2.1.2 Analyse du discours .....	89
4.2.2.2 Les mesures de Rappel/Précision et l'évaluation dans MUC.....	90

4.2.2 Portabilité et Performance .....	90
4.3. Recherche de l'information .....	91
4.3.1 Tâches classiques en RI.....	92
4.3.1.1 Requêtes Ad-hoc, Routage et Filtrage.....	92
4.3.1.2 Classification de Texte.....	93
4.3.2 Analyse automatique du texte et Architecture d'un système de RI.....	94
4.3.2.1 Indexation.....	95
4.3.2.2 Architecture d'un système de RI .....	96
4.3.3 Les Stratégies de Recherche en RI.....	96
4.3.3.1 Modèles Booléens .....	97
4.3.3.2 Modèles Vectoriels.....	97
4.3.3.3 Modèles Probabilistes .....	99
4.3.4 Techniques de Recherche Interactives et retour utilisateur.....	101
4.3.4.1 Formalisme mathématique du feedback.....	102
4.3.4.2 Ré-estimation des poids des composantes de la requête .....	102
4.3.4.3 Expansion automatique de la requête et réestimation des poids de la requête .....	103
4.3.5 Evaluation .....	103
4.3.5.1 Le Graphe de Précision et de Rappel .....	103
4.3.5.2 Autres mesures de performance .....	105
4.4 Conclusion.....	105
4.5 Bibliographie .....	107

## **Chapitre 5. Extraction de passages 113**

5.1 Introduction .....	113
5.2 Segmentation de Texte .....	114
5.2.1 Introduction.....	114
5.2.2 Quelques algorithmes pour la segmentation .....	114
5.2.2.1 Segmentation par calcul de la similarité.....	115
5.2.2.2 Cohésion contextuelle et lexicale.....	118
5.2.2.3 D'autres approches pour la segmentation.....	119
5.2.3 Evaluation des systèmes de segmentation.....	120
5.3 Résumé de Texte .....	120
5.3.1 Introduction.....	120
5.3.2 Quelques algorithmes pour le résumé automatique de texte .....	121
5.3.2.1 Résumé basé sur des mesures de similarité.....	122
5.3.2.3 Résumé par des méthodes d'apprentissage .....	124
5.3.3 Génération de résumés extraits à partir de résumés manuels.....	126
5.3.4 Evaluation et perspectives .....	128
5.4 Conclusion.....	129
5.5 Bibliographie .....	130

## **Partie III : Nouveaux modèles**

### **Chapitre 6. Modèles de séquences pour l'extraction d'information avec des requêtes fermées 138**

6.1 Introduction .....	138
6.2 Analyse de séquences .....	139
6.2.1 Formalisme.....	139
6.2.2 Caractérisation de séquences.....	140
6.2.2.1 Le codage continu .....	140
6.2.2.2 Information Morpho-syntaxique .....	140
6.2.2.3 Sélection de Variable .....	141
6.3 Modèles de Séquences.....	142
6.4 Experiences.....	144
6.4.1 Corpus .....	144
6.4.2 Extraction et Surlignage.....	145

6.4.3 Grammaires .....	146
6.4.4 Evaluation .....	147
6.4.4.1 Comparaison des représentations $U$ et $(U, NP, N, ADJ, V)$ .....	147
6.4.4.2 Comparaison des modèles.....	148
6.4.5 Validation.....	150
6.4.5.1 Validation des performances.....	150
6.4.5.2 Intervalle de Confiance t-Bootstrap par classe.....	151
6.5 Conclusion.....	152
6.6 Bibliographie .....	154
<b>Chapitre 7. Apprentissage interactif, semi-supervisé et non-supervisé pour le résumé de texte</b> .....	<b>157</b>
7.1 Introduction .....	158
7.2 Modèle interactif pour le résumé de textes.....	159
7.2.1 Un système de base pour effectuer du résumé automatique : extraction de phrases en utilisant des mesures de similarité .....	159
7.2.2 Apprentissage interactif.....	160
7.3 Modèle automatique basé sur l'apprentissage non-supervisé et semi-supervisé pour le résumé .....	164
7.3.1 Algorithme CEM dans le cas semi-supervisé.....	166
7.3.2 Fonctions Discriminantes .....	1677
7.3.2.1 Règle de décision Bayésienne pour des populations normales .....	1688
7.3.2.2 Régression linéaire .....	1688
7.3.2.3 Discrimination logistique .....	1699
7.3.3 Modèles discriminants et apprentissage non-supervisé et semi-supervisé.....	170
7.3.3.1 Algorithme CEM-regression.....	17070
7.3.3.2 Algorithme CEM-logistique.....	1722
7.4 Base de Données utilisée pour les expériences .....	1766
7.5 Stratégie d'apprentissage.....	1777
7.5.1 Espace de représentation .....	1777
7.5.2 Apprentissage interactif.....	1777
7.5.3 Apprentissage semi-supervisé et non-supervisé avec les algorithmes CEM-égression et CEM-logisitque .....	1777
7.6 Evaluation.....	1788
7.6.1 Apport de l'interaction .....	17880
7.6.2 Apprentissage semi-supervisé et non-supervisé vs. Système de base.....	18080
7.6.3 Comparaison des algorithmes CEM-régression et CEM-logistique .....	1822
7.7 Conclusion.....	1844
7.8 Bibliographie .....	1866
<b>Discussion</b> .....	<b>189</b>
<b>Annexe</b> .....	<b>191</b>
<b>Bibliographie Personnelle</b> .....	<b>201</b>
<b>Liste des Figures</b> .....	<b>203</b>
<b>Liste des Tables</b> .....	<b>207</b>
<b>Liste des Algorithmes</b> .....	<b>209</b>



# Introduction

## 1. Apprentissage et information textuelle

L'accès à l'information textuelle a motivé depuis de nombreuses années les travaux de chercheurs issus de différentes communautés comme les linguistes, les informaticiens et les statisticiens. Les différents courants issus de ces communautés se sont concentrés sur un ensemble de problématiques spécifiques et ont créé des domaines scientifiques qui ont rapidement évolué de façon autonome. C'est par exemple le cas de la Recherche d'Information (RI), de l'Extraction d'Information (EI), ou dans le cas des statisticiens, des réponses aux questionnaires, des analyses stylistiques, etc.

Ces dernières années, le domaine de l'accès à l'information textuelle a connu une évolution rapide, avec en particulier le développement de grandes bases de données textuelles et du web. Les frontières qui s'étaient dessinées entre les différents domaines traditionnels du texte sont actuellement largement re-dessinées pour créer un grand domaine que nous désignons ici par "accès à l'information textuelle". De nouvelles problématiques apparaissent auxquelles les différentes communautés essaient d'apporter des réponses en adaptant leurs outils ou en développant de nouveaux. En particulier, il est devenu important d'être capable de traiter d'énormes quantités de données textuelles, d'apporter des solutions diversifiées aux nouvelles demandes des utilisateurs, et d'automatiser les outils qui permettent d'exploiter l'information textuelle.

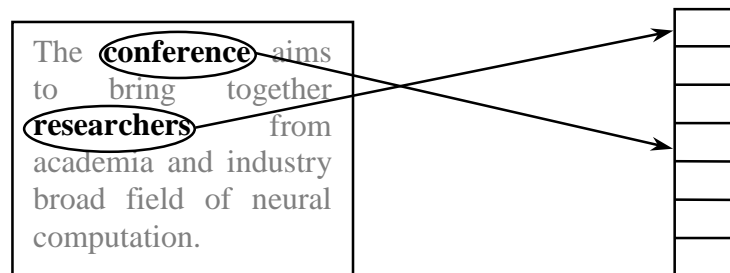
L'apprentissage automatique propose une gamme d'outils qui permettent d'avancer dans ces directions. C'est dans ce cadre que se situe notre travail qui vise à explorer le potentiel des techniques d'apprentissage pour répondre aux besoins de recherche et d'analyse de l'information textuelle.

Ce n'est que récemment que la communauté apprentissage a commencé à appliquer ses outils au texte. Les chercheurs en apprentissage sont très souvent venus au texte en penchant sur les problèmes soulevés par l'accès à l'information sur le *web* ou dans de grosses bases de données. L'intérêt que soulève l'étude de ces problèmes a incité la communauté apprentissage à travailler sur l'automatisation de différents traitements utilisés pour une gamme de tâches allant de la **RI** à l'**EI**. Le but est de concevoir, autant que faire se peut, des méthodes indépendantes du domaine spécifique traité ou du corpus.

Les techniques en apprentissage automatique s'appuient sur l'hypothèse qu'il est possible d'effectuer de nombreuses tâches pratiques de traitement de l'information textuelle par des analyses assez frustes du texte. Très souvent, elles n'utilisent que peu ou pas du tout d'information issue du traitement de la langue. Cette hypothèse est bien sûr très forte, mais est confortée par la nature de nombre de tâches classiquement traitées en RI et également par la nature de certains corpus comme par exemple les forums de discussion ou les e-mails dont la structure grammaticale ou même l'orthographe sont souvent approximatives. Dans notre travail, nous avons employé des méthodes issues de l'apprentissage numérique pour des tâches d'analyse textuelle que l'on peut qualifier de "bas niveau".

## 2. Questions abordées

La grande taille des collections de documents, ainsi que la variabilité et la complexité des informations textuelles, ne permettent généralement pas d'utiliser une représentation sophistiquée des documents, ni des modèles complexes pour traiter automatiquement et rapidement de grosses masses de données. Pour ces raisons, les modèles de l'apprentissage numérique ont adopté l'approche classique « sac de mots » (*bag of words*) pour représenter un document, chaque document étant codé par exemple par l'histogramme de ses mots clés<sup>1</sup> :



Représentation sac de mots

Cette représentation rudimentaire bien que largement utilisée dans le domaine de la RI ne permet pas de "comprendre" le sens des textes. Par contre elle présente l'avantage d'être facilement manipulable et exploitable par des techniques statistiques génériques. « *Est il possible d'aller plus loin avec ces techniques statistiques et de réaliser des tâches un peu plus sophistiquées que celles traditionnellement abordées en recherche d'information ?* » C'est la première question que nous avons abordée dans nos travaux. Pour cela, nous avons cherché à utiliser des techniques permettant d'exploiter automatiquement l'information "séquence de mots" présente dans le texte et que rend inexploitable la représentation sac de mots.

Un autre problème que suscite l'automatisation du traitement de grandes quantités de données et le développement d'outils génériques, est la constitution et l'exploitation des bases de données nécessaires pour apprendre. Alors que les données sont extrêmement

<sup>1</sup> Salton G. et McGill M.J. (1983), Introduction to Modern Information Retrieval. *McGraw-Hill*, New York, NY.

abondantes, la constitution de bases de données étiquetées nécessaires à l'apprentissage supervisé est extrêmement coûteuse et longue. Ceci est préjudiciable au développement rapide de systèmes génériques permettant avec un minimum d'interaction le traitement de différents corpus et capable de s'adapter en continu aux flux observés. Pour réduire ce temps de développement et exploiter les informations disponibles sans passer par une phase coûteuse d'ajout de connaissances, l'utilisation des données non-étiquetées est devenue primordiale. C'est ainsi que récemment, est né dans la communauté apprentissage le paradigme de l'apprentissage semi-supervisé qui a pour but d'apprendre à effectuer des tâches génériques de l'apprentissage supervisé tout en exploitant de petites quantités de données étiquetées, simultanément à de grandes masses de données brutes, i.e. non-étiquetées. C'est le deuxième problème que nous avons abordé avec comme support la tâche de résumé automatique.

### **3. Modèles de séquences**

Pour répondre à la première question, nous proposons d'utiliser des modèles dynamiques de séquences. L'analyse de séquences de mots est faite au moyen de modèles stochastiques, tels que des Modèles de Markov Cachés (MMC) et des Perceptrons Multi-Couches (PMC). Un avantage de ce formalisme est que nos modèles de séquences manipulent différentes sources d'informations, continues ou discrètes, et qu'il est possible de traiter avec un même modèle générique différentes tâches d'accès à l'information.

Manipuler des modèles de séquences nécessite une représentation des mots en petite dimension. Un constat surprenant que nous avons fait expérimentalement est que des représentations très simples des séquences de mots sont suffisantes pour des tâches relativement complexes. Nous introduisons donc une telle représentation pour coder les documents textuels sous forme de séquences de mots, qui est compatible avec l'utilisation des modèles numériques. Nous étudions également l'impact de l'ajout d'informations morpho-syntaxiques sur cette représentation. Pour exploiter ces modèles, nous proposons d'introduire différentes contraintes grammaticales sur leurs sorties, nous étudions quelques exemples de ces contraintes. Pour tester nos modèles nous avons utilisé des articles du Wall Street journal du corpus MUC-6<sup>2</sup>. Les deux tâches que nous avons considérées sont des tâches élémentaires de l'extraction d'information, elles consistent respectivement i) à surligner toutes les descriptions de changement d'événements personnels (des offres d'emploi, des démissions) ; dans ce cas, notre but est d'étiqueter des sous-séquences de texte comme pertinentes, ou non, pour la tâche, et ii) à extraire des informations comme le nom et la position de la personne concernée, auquel cas, les sous-séquences devront être étiquetées comme Personne, Position ou non-pertinente.

---

<sup>2</sup> MUC-6, (1996), Proceedings of the sixth message understanding conference, *ed, Morgan Kaufmann*.

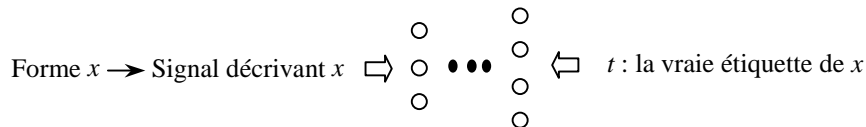
#### 4. Apprentissage semi-supervisé

L'*apprentissage semi-supervisé* a principalement été exploré dans la communauté apprentissage pour des tâches génériques de discrimination.

Pour apprendre la relation entre les exemples et leur étiquette, l'apprentissage supervisé propose d'optimiser un critère spécifique qui met en relation ces étiquettes avec les sorties du modèle calculées sur une représentation décrivant les exemples. Le critère théorique le plus utilisé pour la discrimination est l'espérance de l'erreur de classification (Tsyarkin)<sup>3</sup>. Cette relation peut se schématiser par :

##### Apprentissage Supervisé

- On apprend la densité jointe  $p(x,t)$  ou la probabilité conditionnelle  $p(t/x)$



En contrepartie, dans le cadre de l'apprentissage non-supervisé, on ne possède pas pour les données non-étiquetées l'information de sortie désirée.

La démarche la plus couramment employée consiste alors à passer par une phase de modélisation des données où l'on essaie de modéliser simultanément la structure interne des données issue de la distribution marginale  $p(x)$  pour les données non-étiquetées et la densité jointe  $p(x,t)$  pour les données étiquetées. On adopte alors une démarche typique de l'apprentissage non supervisé et l'on essaie d'introduire et de traiter une information supplémentaire concernant les exemples non-étiquetés. Pour modéliser les  $p(x)$ , on fait souvent des hypothèses sur la forme de la distribution des données, i.e. on utilise des modèles paramétriques. Les quantités estimées  $p(x)$  et  $p(x,t)$  seront ensuite utilisées pour calculer les probabilités conditionnelles  $p(t/x)$  qui serviront de base à la décision.

Une approche alternative consiste à adapter à l'apprentissage semi-supervisé les techniques discriminantes classiques en supervisé. Deux idées principales ont été exploitées pour cela.

La première a été mise au point dans le cadre du traitement de signal adaptatif : il s'agit d'utiliser dans un contexte non-supervisé les sorties prédites par le système lui-même pour construire des sorties désirées. Celles-ci seront ensuite utilisées pour apprendre à partir d'une technique supervisée. Cette approche est connue sous le nom de *décision-dirigée*.

La deuxième idée, qui est, elle, issue de la communauté apprentissage, repose sur l'utilisation simultanée de deux classifieurs. Ceux-ci jouent alternativement le rôle de maître et d'élève dans un algorithme d'apprentissage itératif : la sortie calculée par l'un sera prise comme sortie désirée par l'autre et réciproquement, jusqu'à la convergence

<sup>3</sup> Y. Tsyarkin. Foundations of the Theory of Learning Systems. *Mathematics in science and engineering*. Academic Press. Vol. 101, 1973.

éventuelle de l'ensemble. Le critère d'apprentissage est ici d'optimiser la cohérence entre les deux classifieurs. Cette approche est connue sous le nom d'*auto-supervision*. Dans le cadre de l'apprentissage semi-supervisé, cette idée a été reprise, très récemment, dans des algorithmes plus sophistiqués comme le *Co-Boosting* et le *Co-Training*. Le reproche que l'on peut faire à ce genre de modèle est que, pour apprendre, il faut utiliser plusieurs modalités représentant une même donnée, ce qui limite considérablement l'emploi de ces systèmes.

Nous proposons un système semi-supervisé discriminant basé sur une idée similaire à celle de décision dirigée où le modèle apprend d'après sa propre sortie. Comme critère d'apprentissage nous proposons une adaptation du critère de "vraisemblance de classification" qui a été proposé dans le cadre de l'apprentissage non supervisé pour faire du "clustering". Notre critère traite les vecteurs indicateurs de classes comme des paramètres manquants dans le cas d'exemples non-étiquetés. Ce système utilise un classifieur de base qui peut être choisi dans toute la gamme des techniques numériques de discrimination. Nous présentons des analyses et des preuves de convergence dans deux cas particuliers qui sont d'une part un classifieur linéaire et d'autre part un classifieur logistique. Notre motivation pour le développement de ces techniques a été de construire des systèmes de résumé automatique par extraction de phrases du texte original. Nous présentons plusieurs instanciations des idées issues de l'apprentissage semi-supervisé pour cette tâche.

## 5. Plan de la thèse

La thèse est composée de trois grandes parties : les deux premières parties introduisent les domaines étudiés ainsi que les outils qui seront utilisés, la troisième partie expose notre contribution.

- La première partie est consacrée aux modèles de l'apprentissage et comporte trois chapitres. Au chapitre 1, nous présentons les modèles de l'apprentissage numérique que nous avons employés dans notre travail. Ce chapitre introduit ainsi les bases des réseaux de neurones, des modèles de Markov cachés et des machines à vecteurs supports. Le chapitre 2 introduit les méthodes d'optimisation (les algorithmes) principales utilisées en apprentissage supervisé et non supervisé. Ces deux chapitres n'ont pas de prétention à l'exhaustivité, au contraire, nous nous sommes limités à introduire les notions qui sont nécessaires pour les travaux exposés dans les chapitres 6 et 7. Le chapitre 3 est une bibliographie sur l'apprentissage semi-supervisé qui est centrale à l'étude présentée au chapitre 7. Cette synthèse est plus personnelle que celle des deux chapitres précédents car ces travaux sont bien plus récents.
- La partie 2, introduit des bases de l'accès à l'information textuelle. Elle comporte deux chapitres. Le chapitre 4 donne un aperçu des modèles et techniques classiques utilisés en **RI** et en **EI**. Là aussi, nous nous contentons d'exposer des techniques de base, ceci à l'intention du spécialiste d'apprentissage qui ne connaît pas le domaine

du texte. Le *chapitre 5* présente quant à lui les méthodes de l'extraction de passages utilisées en segmentation et en résumé de texte. Ces méthodes sont plus récentes et elles sont plus ciblées que celles présentées au chapitre précédent.

- Dans la troisième partie nous présentons deux contributions originales. Le *chapitre 6* présente le système stochastique pour l'analyse de séquences et l'extraction d'information dans les textes. L'idée novatrice -pour l'analyse statistique de texte- est de ne plus considérer le texte comme un ensemble non-ordonné de termes mais comme une *séquence* de mots. Pour cela nous proposons sous un même formalisme deux modèles de séquences qui permettent de travailler à un niveau plus fin que les modèles classiques de recherche d'information. Ce système est testé sur des tâches d'extraction d'information de surface qui sont inspirée des compétitions MUC. Le *chapitre 7*, présente deux approches permettant d'introduire l'apprentissage dans les systèmes de résumé automatique où la tâche de résumé est vue sous l'angle de l'extraction de phrases pertinentes d'un document. La première approche est basée sur l'apprentissage interactif : le système utilise le jugement d'un utilisateur pour corriger ses erreurs de décisions. La deuxième approche est basée sur l'apprentissage non-supervisé et semi-supervisé et elle va plus loin vers l'automatisation des systèmes de résumé. En annexe, nous décrivons le Système d'Aide au Résumé Automatique (S.A.R.A.) que nous avons développé et qui opère aussi bien en mode interactif qu'en mode totalement automatique.

## Notation

$x \in X$	l'entrée et l'espace des entrées
$y \in Y$	la sortie et l'espace des sorties
$t \in T$	L'étiquette d'un exemple et l'espace des étiquettes
$C$	Cluster ou classe
$P$	Une partition
$c$	Le nombre de clusters ou de classes
$p$	dimension de l'espace de représentation des exemples
$n$	Le nombre d'exemples étiquetés
$m$	Le nombre d'exemples non étiquetés
$D_l = \{(x_i, t_i) / i=1, \dots, n\}$	L'ensemble des exemples étiquetés
$\Psi = \{(k, t) / k \in \{1, \dots, n\}, t \neq t_k\}$	L'ensemble des exemples mal classés.
$D_u = \{x_i / i=1, \dots, m\}$	L'ensemble des exemples non étiquetés
$H$	La fonction caractéristique de $\mathbb{R}^+$ ou la fonction Heaviside
$F$	La fonction de transfert
$\eta$	Le pas d'apprentissage
$S$	L'ensemble des états d'un modèle de Markov caché
$\Xi$	l'espace des caractéristiques
$\Phi : X \rightarrow \Xi$	Fonction qui plonge l'espace des données $X$ dans $\Xi$
$K$	La fonction noyau $K(x, y) = \Phi(x) \cdot \Phi(y)$
$s$	Le vecteur support
$N_s$	Le nombre de vecteurs support
$L_p$	Le lagrangien primal
$w$	Les paramètres du modèle
$M_w$	Le modèle de paramètres $w$
$p(x, t)$	La probabilité jointe entre les entrées et les étiquettes
$E_{x,t}[\cdot]$	L'espérance mathématique associé aux variables aléatoires $x$ et $t$
$E_D[\cdot]$	L'espérance sur l'ensemble $D$ de données
$\nabla$	le gradient par rapport aux paramètres $w$
$B$	Le nombre d'échantillons bootstrapés
$I$	Le nombre d'itération
$\pi$	La probabilité a priori des classes
$\theta$	Les paramètres d'un modèle de densité de mélange
$p(x/C, \theta)$	La densité conditionnelle des classes
$p(C/x)$	La probabilité a posteriori des classes
$L_M$	Le logarithme de la vraisemblance de mélange
$L_C$	Le logarithme de la vraisemblance de classification
$\Sigma$	La matrice de la covariance
$\mu$	Le vecteur de la moyenne
$\mathcal{N}_p(\mu, \Sigma)$	La distribution normale dans l'espace $\mathbb{R}^p$
det	Le déterminant

$\bar{x}$	La moyenne des données
Var	La variance des données
$I_p$	La matrice identité de l'espace vectoriel de dimension $p$
$Cl$	Un classifieur
$q$	La requête
$P_q$	Pertinence par rapport à la requête
$\varphi$	Une phrase
$\beta$	Paramètre d'un modèle logistique



## **Partie I**

### **Modèles dynamiques et apprentissage numérique**



# Chapitre 1

## Modèles de l'apprentissage numérique

**Résumé.** Dans ce chapitre nous allons présenter les modèles utilisés dans notre étude, à savoir les réseaux de neurones, les modèles de Markov cachés et les machines à vecteurs support. Nous allons donner une description détaillée de ces trois modèles, leurs hypothèses de base, et les algorithmes d'apprentissage associés. Nous verrons dans les chapitres suivants (6 et 7) que mis à part leur capacité d'apprentissage et leur robustesse, ces modèles peuvent être intégrés dans des systèmes de recherche d'information ou de fouille de données textuelles plus complexes.

**Mots clés :** Apprentissage numérique, réseaux de neurones, modèles de Markov cachés, Machines à vecteurs Support.

### 1.1 Introduction

Pour traiter l'information, on dispose en général de deux types de connaissances :

- D'une part, on peut avoir recours à des connaissances *a priori* ou à des modélisations pour automatiser une tâche pour laquelle on manque d'étiquetages explicites des formes (les techniques s'appuyant sur ce type de connaissance sont connues sous le nom de *l'apprentissage non-supervisé* et elles sont introduites au chapitre 2).
- D'autre part, on dispose de systèmes exploitant l'information *a posteriori* (e.g. étiquetage des formes) et qui possèdent la faculté *d'apprendre* (associer les formes d'entrées à leur sorties). Ces systèmes font l'économie d'un grand travail de modélisation et ils sont capables de s'adapter ce qui n'est pas le cas des systèmes traitant l'information *a priori* qui sont eux figés.

La plupart des techniques permettant d'exploiter les connaissances *a posteriori* sont des techniques numériques. Elles ont été très utilisées pour résoudre des problèmes variés allant de la commande en robotique à la reconnaissance des formes.

Nous allons présenter dans ce chapitre des modèles de ce type, en nous limitant à ceux que nous avons utilisés dans notre travail. Ces modèles sont : les réseaux de neurones (paragraphe 2), les modèles de Markov cachés (paragraphe 3) et les machines à vecteurs

support (paragraphe 4). Pour chaque modèle, nous détaillerons les hypothèses de fonctionnement ainsi que les différentes architectures permettant leur mise en œuvre.

## 1.2 Réseaux de neurones

### 1.2.1 Introduction

Les premières études sur les neurones artificiels remontent aux travaux de McCulloch et Pitts [McP43] qui ont essayé de déterminer les propriétés de ces neurones à partir de celles des neurones du système nerveux. Ils ont commencé à étudier les *neurones binaires* ne reproduisant que des sorties 0 ou 1. Le résultat important de leurs travaux est que ces neurones binaires sont capables de calculer n'importe quelle fonction logique, ce qui les rend équivalent à une machine de Turing universelle [TUR50].

Vers la fin des années quarante, d'autres chercheurs ont essayé d'expliquer les mécanismes de l'apprentissage, de la mémoire et du conditionnement à partir de groupes de cellules. Par exemple Hebb a expliqué dans sa *loi de Hebb* les processus mis en jeu lors de l'apprentissage, en fonction de l'expérience [HEB49]. Il proposait pour cela un modèle où les cellules apprennent à modifier l'intensité des connexions qui les relient en fonction de leur activité simultanée. Au même moment Minsky [MIN52] a conçu *SNARC* qui fut le premier modèle réel d'un réseau de neurones. *SNARC* était capable d'apprendre en corrigeant l'intensité de ses synapses. Sur cette base, Rosenblatt [ROS58] créa *Mark I*, le premier perceptron qui avait pour but de faire apprendre des catégorisations perceptives à un réseau de neurones artificiel. Dans les mêmes années Widrow et Hoff [WiH60] ont présenté une loi d'apprentissage proche de celle du perceptron et ont construit l'*ADALINE* qui apprenait à atténuer le bruit des lignes téléphoniques. La conclusion de tous ces travaux fut l'ouvrage *learning machines* de Nilsson [NIL65] qui posa les fondations mathématiques de l'apprentissage automatique pour la reconnaissance des formes. Pendant toute cette période, la recherche sur les réseaux de neurones s'est faite sans une réelle mise en question des méthodes employées et des résultats obtenus.

Vers la fin des années soixante, la communauté de l'apprentissage numérique a commencé la synthèse de ces recherches. Minsky et Papert [MiP69] ont montré les limites de ces machines. Par exemple un perceptron n'est pas capable d'apprendre la fonction logique XOR, parce que, contrairement à la fonction ET, elle n'est pas linéairement séparable (Figure 1). Pendant les années de désintérêt qui ont suivi, plusieurs auteurs ont exploré le sujet. Anderson et Rosenfeld [AnR88] ont réuni dans leur recueil un bon nombre d'articles correspondant à cette période. Puis, à la fin des années quatre-vingt, le domaine a connu un nouvel essor. Hopfield [HOP84] d'un côté, Rumelhart et McClelland [RuM86] de l'autre, ont montré que certaines règles d'apprentissage (par exemple la technique de *retro-propagation de l'erreur*) pouvaient permettre à des réseaux de neurones d'apprendre des fonctions que le perceptron ne pouvait apprendre. Il était aussi devenu clair que l'approche purement symbolique en intelligence artificielle ne permettait pas à elle seule de résoudre certains problèmes et que les réseaux de neurones offraient une approche complémentaire.

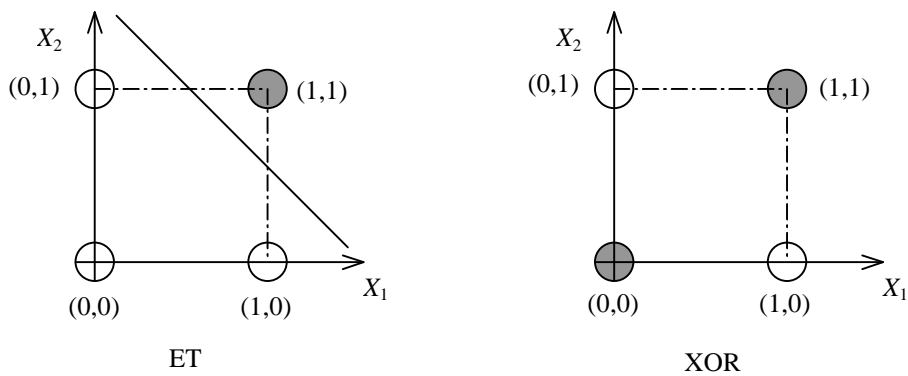


Figure 1. La fonction ET (à gauche) est linéairement séparable mais pas la fonction XOR (à droite).

Pendant les années quatre-vingt dix, les réseaux de neurones étaient devenus un domaine attractif réunissant autour d'un même thème des psychologues, des physiciens, des mathématiciens, des biologistes et des informaticiens.

### 1.2.2 Unités neuronales

Un réseau de neurones est un ensemble d'unités élémentaires reliées entre elles. Dans ce qui suit nous allons présenter succinctement les unités élémentaires les plus utilisées.

#### 1.2.2.1 Unités à seuil

Le modèle de McCulloch et Pitts [McP43] s'appuie sur une formulation simple d'un neurone possédant des entrées binaires ou réelles et une sortie binaire. Sa fonction consiste à effectuer une somme pondérée des entrées. La pondération est modélisée par les coefficients synaptiques du modèle. Si cette somme est supérieure à un seuil  $\theta$ , la sortie vaut 1 et dans le cas contraire elle vaut 0 :

$$y = H(w^t x - \theta) \quad (1.1)$$

Avec  $H(\cdot)$  la fonction caractéristique de  $\mathbb{R}^+$  ou la fonction de *Heaviside* (figure 3, a),  $x$  le signal d'entrée et  $w^t$  le transposé du vecteur des poids. La figure 2 illustre ce modèle.

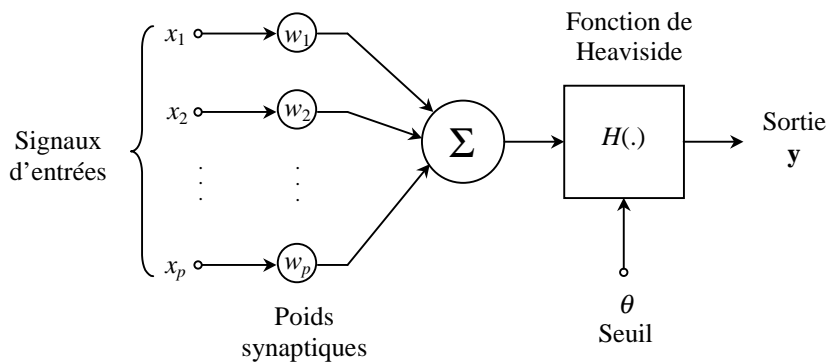


Figure 2. Modèle linéaire des réseaux de neurones.

Ce modèle à seuil sépare l'espace d'entrée en deux demi-espaces par l'hyperplan d'équation :

$$w^t x - \theta = 0$$

### 1.2.2.2 Unités linéaires et sigmoïdes

D'autres fonctions de transfert ont également été proposées afin d'obtenir des solutions réelles et non plus binaires. Elles sont toutes de la forme :

$$y = F(w^t x - \theta) \quad (1.2)$$

Dans laquelle  $F$  est une fonction à valeurs réelles. Les formes les plus généralement employées sont ;

- des fonctions linéaires :

$$F(x) = \alpha x \text{ avec } \alpha \text{ un réel positif non nul.}$$

- des fonction linéaires par morceaux, Figure 3 (b) :

$$F(x) = \begin{cases} 1, & \text{si } x \geq 0.5 \\ x + 0.5, & \text{si } -0.5 < x < 0.5 \\ 0, & \text{sinon} \end{cases}$$

- ou des fonctions sigmoïdes, Figure 3 (c) :

$$F(x) = \frac{1}{1 + \exp(-\kappa x)}$$

Où  $\kappa$  représente la pente de la sigmoïde. Avec différentes valeurs de  $\kappa$  on obtient des sigmoïdes de différentes pentes.

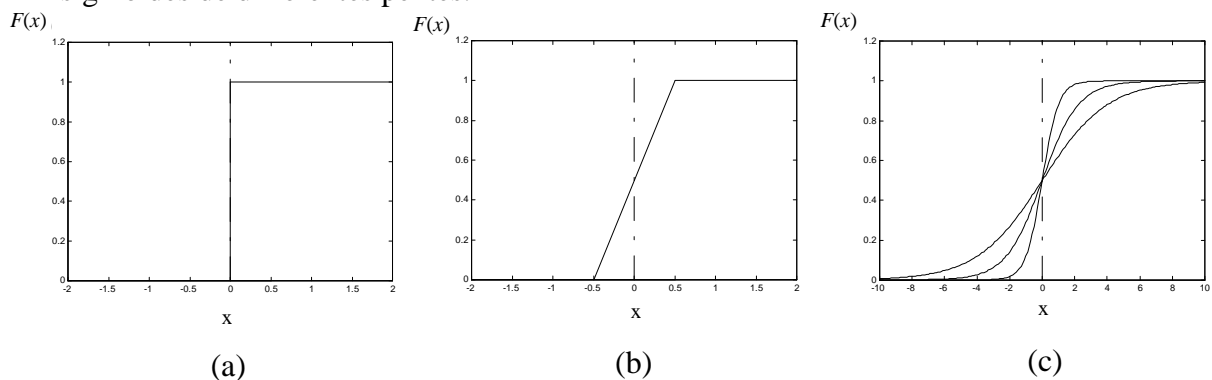


Figure 3. Différentes fonctions de transfert (a) Fonction de Heaviside, (b) Fonction linéaire par morceaux, (c) Fonction sigmoïde.

D'autres types d'unités ont été proposés comme les unités *radiales* qui reposent sur un calcul de distance, les unités *produits* qui amplifient les sommes pondérées des entrées par des produits de ces entrées et de nombreuses autres unités.

### 1.2.3 Algorithmes d'apprentissage

Dans ce qui suit, nous allons nous intéresser à deux règles locales d'apprentissage pour les réseaux de neurones dans le cas supervisé, i.e. la règle de *Hebb* et la règle de *Widrow-Hoff*. On emploie le terme « règle locale » dans le sens où chaque cellule de sortie apprend sans avoir besoin de connaître la réponse des autres cellules. Seule importe l'information qui lui est apportée en entrée.

#### 1.2.3.1 Règle de Hebb

La règle de Hebb est inspirée par les travaux de [HEB49]. C'est la règle la plus simple pour changer les poids synaptiques  $w_{ji}$ ; elle s'exprime par le produit des signaux d'entrée et de sortie. Cette règle consiste à renforcer d'autant plus un poids entre les cellules d'entrée et de sortie que cette entrée est corrélée avec la sortie, elle s'écrit :

$$\Delta w_{ji}^{(\tau)} = \eta \cdot y_j \cdot x_i$$

où  $\Delta w_{ji}^{(\tau)}$  est la correction du poids synaptique entre la  $i^{\text{ème}}$  cellule d'entrée et la  $j^{\text{ème}}$  cellule de sortie au temps  $\tau$ ,  $y_j$  est la réponse de la  $j^{\text{ème}}$  cellule de sortie,  $x_i$  est l'entrée de la  $i^{\text{ème}}$  cellule et  $\eta$  est le pas d'apprentissage comprise entre 0 et 1.

Avec cette équation, l'application répétée du signal d'entrée entraîne une augmentation linéaire du poids synaptique  $w_{ji}$ .

#### 1.2.3.2 Règle de Widrow-Hoff

Cette règle est très utilisée avec des modèles tels que le perceptron et l'Adaline (section 1.2.4.1). Tout d'abord, une cellule ne modifie l'intensité de ses synapses, ou n'apprend, que lorsqu'elle se trompe. Si la cellule de sortie est active/inactive alors qu'elle devrait être inactive/active, alors elle diminue/augmente l'intensité des synapses correspondant aux cellules de rétine qui sont actives. Cela revient à diminuer/augmenter l'intensité de la stimulation à la cellule de sortie.

L'ensemble des stimuli à apprendre est présenté dans un ordre aléatoire. Si après une première présentation de l'ensemble des stimuli le modèle commet encore des erreurs, on présente de nouveau l'ensemble des stimuli. La procédure est itérée jusqu'à ce que le modèle soit capable de donner toutes les réponses correctes. Avec les mêmes notations que précédemment, la règle d'apprentissage de *Widrow-Hoff* s'écrit :

$$\Delta w_{ji}^{(\tau)} = \eta (t_j - y_j) \cdot x_i^4$$

où  $t_j$  est la désirée de la  $j^{\text{ème}}$  cellule de sortie.

---

<sup>4</sup> La mise à jour des poids est donc :  $w_{ji}^{(\tau+1)} = w_{ji}^{(\tau)} + \eta (t_j - y_j) \cdot x_i$

## 1.2.4 Quelques réseaux

Dans ce paragraphe, nous présentons quelques modèles de réseaux de neurones appris classiquement avec un algorithme du type *Widrow-Hoff*. Nous commençons par présenter deux modèles très simples, le perceptron et l'adaline, puis nous présenterons le perceptron multi-couches, sans doute le plus utilisé des réseaux de neurones.

### 1.2.4.1 Perceptron et Adaline

Les réseaux à une seule couche et unités d'activation à seuil ont été étudiés par Rosenblatt [ROS58] au début des années soixante. Il appela ces réseaux des perceptrons car ils ont été conçus à l'origine en s'inspirant du système visuel. Ces réseaux ont été appliqués à des tâches de classification dont la plupart concernaient des images binaires. Les propriétés des perceptrons sont reprises en détail dans [BLO62] et [BIS95]. Nous allons exposer brièvement les principes de base de ce modèle dont la compréhension permet de mieux appréhender le fonctionnement des perceptrons multi-couches qui ont été largement utilisés dans nos applications.

#### *Perceptron*

Dans le paragraphe précédent, nous avons noté que les réseaux à une seule couche disposaient de capacités limitées. Pour pallier ces lacunes, Rosenblatt a utilisé des fonctions linéaires dites « d'association » pour transformer les signaux d'entrée (Figure 4). Ces fonctions ont des poids reliés d'une manière aléatoire à un sous-ensemble des entrées du réseau (i.e. pixels de la rétine). La sortie calculée par le perceptron pour une entrée  $x$  est égale à :

$$y = H(w^t \cdot \Phi) = H\left(\sum_{j=1}^M w_j \cdot \Phi_j(x)\right)$$

où  $\Phi = (\Phi_1, \Phi_2, \dots, \Phi_M)$  représente le vecteur d'association et  $w = (w_1, w_2, \dots, w_M)$  est le vecteur des poids synaptiques.

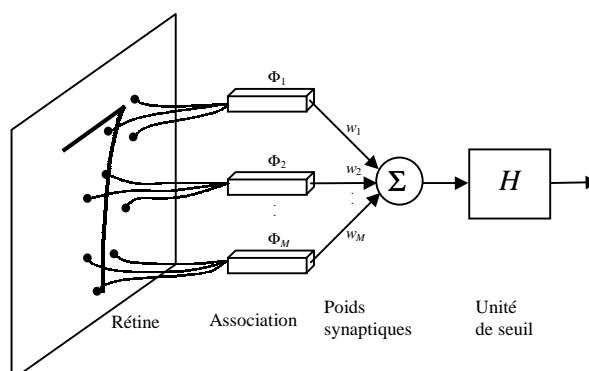


Figure 4. Architecture d'un perceptron composé de quatre composantes principales : la rétine, les fonctions d'association, les poids synaptiques et l'unité à seuil.



L'apprentissage du perceptron est réalisé à l'aide d'un algorithme *en-ligne*<sup>5</sup>. La correction des poids pour une entrée  $x$  s'écrit :

$$\Delta w = \eta \cdot [t(x) - H(w^t \cdot \Phi(x))] \cdot \Phi(x) \quad (1.3)$$

où  $t(x)$  est le désiré de  $x$ ,  $H(w^t \cdot \Phi(x))$  est la sortie correspondant à l'entrée  $x$  et  $\Phi(x)$  est la transformation de  $x$  par l'aire d'association. Cet algorithme converge si les données sont linéairement séparables dans l'espace des  $\Phi$ . Dans ce cas, la convergence du perceptron en un nombre fini d'étapes découle de l'équation 1.3, où les signaux d'entrées sont issus de l'aire d'association et où l'unité d'activation est une unité à seuil. [DuH73, VAP82] proposent par exemple une preuve de cette convergence.

La figure 5 illustre le fonctionnement de l'algorithme d'apprentissage du perceptron pour un problème simple de classification à deux classes [BIS95]. La frontière de décision en pointillés est déterminée par le vecteur de poids  $W^{(0)}$  (Figure 5, a). La direction de la frontière de décision est perpendiculaire à celle des poids synaptiques. Dans le cas où le perceptron se trompe sur un exemple, il corrige son poids en soustrayant l'ancien poids par le vecteur représentant cet exemple (Figure 5, b). En itérant ce processus le perceptron converge et trouve un hyperplan séparateur (Figure 5, c). D'après l'équation (1.3), le changement de poids n'opère que si l'entrée  $x$  est mal classée. L'hyperplan séparateur peut donc être très proche d'un exemple, provoquant ainsi de mauvaises classifications dans le cas d'entrées bruitées (Figure 5, c).

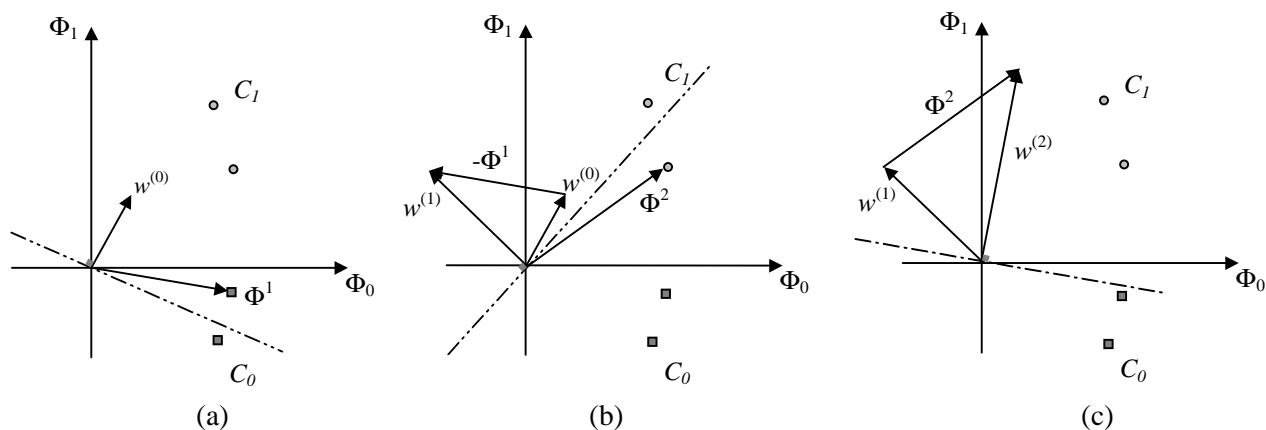


Figure 5. L'illustration de la convergence du perceptron pour un problème simple de classification à deux classes et quatre exemples. a) pour un vecteur de poids, l'hyperplan séparateur est représenté par celui ayant la direction perpendiculaire à ce vecteur, b) la correction de poids se fait en soustrayant le vecteur de poids par le vecteur représentant le premier exemple mal classé, c) en répétant cette procédure le perceptron trouve le bon hyperplan séparateur.

<sup>5</sup> au chapitre 2, section 2.2.2.1, nous allons revenir sur la définition de ce terme

### Adaline

Parallèlement aux travaux de Rosenblat, Widrow et Hoff proposaient l'algorithme de l'Adaline [WiH60]. La règle locale sur laquelle repose cet algorithme optimise un critère quadratique :

$$\Delta w = \eta \cdot (t(x) - w^t \cdot x) \cdot x \quad (1.4)$$

Cette règle est similaire à la règle d'apprentissage du perceptron (1.3). Les différences résident dans la fonction d'activation (fonction Heaviside pour le perceptron) et l'espace d'entrée (l'espace des  $\Phi$  pour le perceptron). La figure 6 illustre sur un exemple de classification à deux classes la différence de comportement entre le perceptron et l'Adaline. En minimisant le nombre d'erreurs, le perceptron trouve l'hyperplan qui sépare au mieux les classes ; ceci au risque d'une mauvaise classification dans le cas de données bruitées. L'Adaline fait des erreurs mais la solution trouvée est généralement plus robuste. De nombreuses variantes ont été proposées pour remédier aux faiblesses reconnues des deux algorithmes.

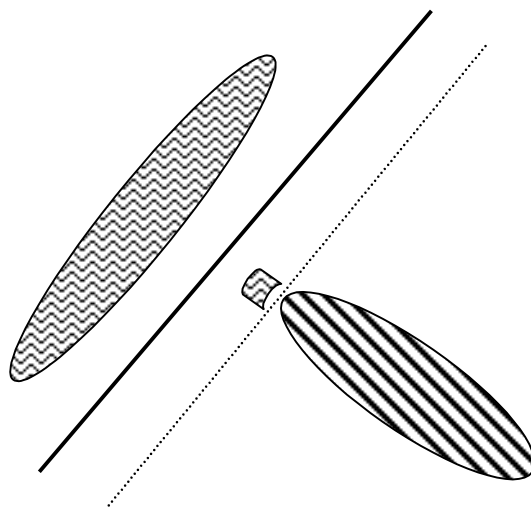


Figure 6. Problème de classification linéaire à deux classes et solutions trouvées par le perceptron (en pointillée) et par l'Adaline (en gras).

#### 1.2.4.2 Perceptrons multi-couches

Vers le milieu des années 80, des systèmes plus puissants permettant d'apprendre des fonctions bien plus complexes que les fonctions linéaires ont été proposés, par exemple des systèmes basés sur des *fonctions radiales* (en anglais *radial basis function*) [REN89], des techniques basées sur des généralisations de la décomposition en valeurs propres et de la décomposition en valeurs singulières [ABD88], et bien d'autres systèmes. Mais aucune de ces approches n'a atteint la popularité des perceptrons multi-couches (PMC) qui constituent une part importante des applications des réseaux de neurones. La figure 7 montre un exemple d'un tel réseau PMC à une couche cachée. Dans ce réseau, il y a  $p$  entrées,  $l$  unités sur la couche cachée et  $c$  unités sur la couche de sortie.

La fonction analytique correspondante à la figure 7 s'écrit comme suit. L'activation de la  $j^{\text{ème}}$  unité de la couche cachée est obtenue d'abord par une composition :

- D'un produit scalaire  $a_j$ , des  $p$  entrées  $\{x_i\}_{i=1..p}$  ainsi que du biais  $x_0=1$  et des poids synaptiques  $\{w_{ji}^{(1)}\}_{i=1..p}$  reliant les entrées à cette unité et des paramètres du biais  $w_{j0}^{(1)}$  :

$$a_j = \sum_{i=1}^p w_{ji}^{(1)} .x_i + w_{j0}^{(1)} = w_{j.}^{(1)T} .x + w_{j0}^{(1)}$$

avec  $w_{j.}^{(1)T}$  qui est le transposé du vecteur poids reliant la  $j^{\text{ème}}$  unité à l'entrée.

- et d'une fonction d'activation  $F(.)$  :

$$z_j = F(a_j)$$

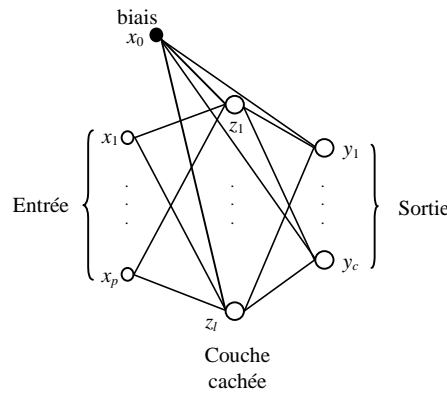


Figure 7. Architecture d'un perceptron multi-couches à une couche cachée. Les paramètres de biais sont introduits par des poids liés à une unité d'entrée supplémentaire ayant la valeur fixée  $x_0=1$ .

De même, l'activation de la  $k^{\text{ème}}$  unité de la couche de sortie est obtenue en transformant le produit scalaire avec la fonction d'activation  $F$  :

$$y_k = F(a_k) = F\left(\sum_{j=0}^l w_{kj}^{(2)} .F\left(\sum_{i=0}^p w_{ji}^{(1)} .x_i\right)\right)$$

Les fonctions d'activation peuvent être celles présentées dans la section 1.2.2.2. Nous nous intéresserons, sans restreindre la généralité de notre discussion, au cas où ces fonctions sont différentiables<sup>6</sup>. De plus nous pouvons étendre l'étude de ces réseaux à ceux à  $N$  couches cachées, où  $N \geq 2$ .

L'algorithme d'apprentissage le plus courant pour les PMC est l'algorithme de *rétro-propagation* qui est une adaptation de l'algorithme de *Widrow-Hoff* à ces réseaux à couches. Différents critères d'erreur peuvent être utilisés, les plus courants étant des

<sup>6</sup> Le cas où les fonctions de transfert ne seraient pas différentiables est repris au chapitre 2, section 2.2.3.3. Dans ce cas, moyennant une condition suffisante, on peut encore appliquer le théorème de la convergence.

mesures de distance ou d'entropie, calculées à partir des sorties désirées et des sorties calculées. Si on prend par exemple le critère quadratique qui est proposé dans l'algorithme original :

$$E_x = \frac{1}{2} \times \|y - t\|^2 = \frac{1}{2} \times \sum_{k=1}^c (y_k - t_k)^2$$

L'algorithme de Widrow-Hoff s'écrit :

$$\Delta w_{ji} = -\eta \cdot \frac{\partial E_x}{\partial w_{ji}} = -\eta \cdot \frac{\partial E_x}{\partial a_j} \times \frac{\partial a_j}{\partial w_{ji}}$$

Pour la correction des poids entre la couche cachée et la sortie on a :

$$\Delta w_{kj}^{(2)} = -\eta \cdot \frac{\partial E_x}{\partial a_k} \times \frac{\partial a_k}{\partial w_{kj}^{(2)}}$$

Où  $\frac{\partial a_k}{\partial w_{kj}^{(2)}} = z_j$ , et  $\frac{\partial E_x}{\partial a_k}$  peut se calculer en appliquant la règle de dérivation des fonctions composées :

$$\frac{\partial E_x}{\partial a_k} = F'(a_k) \cdot \frac{\partial E_x}{\partial y_k} = \varphi'(a_k) \cdot (y_k - t_k)$$

Soit en posant  $\delta_k = F'(a_k) \cdot (y_k - t_k)$ , il vient :

$$\Delta w_{kj}^{(2)} = -\eta \cdot \delta_k \cdot z_j$$

Pour la correction des poids entre la couche cachée et la couche de sortie on a :

$$\Delta w_{ji}^{(1)} = -\eta \cdot \frac{\partial E_x}{\partial a_j} \times \frac{\partial a_j}{\partial w_{ji}^{(1)}}$$

Où,  $\frac{\partial a_j}{\partial w_{ji}^{(1)}} = x_i$ , et  $\frac{\partial E_x}{\partial a_j}$  s'écrit en appliquant la même règle de dérivation que précédemment :

$$\frac{\partial E_x}{\partial a_j} = F'(a_j) \cdot \sum_{k=1}^c \delta_k \cdot w_{kj}^{(2)}$$

Soit en posant  $\delta_j = F'(a_j) \cdot \sum_{k=1}^c \delta_k \cdot w_{kj}^{(2)}$ , il vient :

$$\Delta w_{ji}^{(1)} = -\eta \cdot \delta_j \cdot x_i$$

Ce calcul peut se généraliser dans le cas de réseaux à  $N$  couches cachées. L'algorithme ci-dessous résume l'expression de la mise à jour des poids dans l'algorithme de rétro-propagation du gradient pour un apprentissage en-ligne.

---

$$\Delta w_{ji} = -\eta \cdot \delta_j \cdot x_i$$

Si  $j \in \{\text{unités de sorties}\}$  alors

$$\delta_j = F'(a_j) \cdot (y_j - t_j)$$

Sinon

$$\delta_j = F'(a_j) \cdot \sum_{k \in AV(j)} \delta_k \cdot w_{kj}$$

Où  $AV(j)$  est l'ensemble des unités qui utilisent comme entrée la sortie de  $j$ .

---

Algorithme 1. Algorithme de rétro-propagation du gradient dans le cas de l'apprentissage en-ligne.

## 1.3 Modèles de Markov Cachés

### 1.3.1 Introduction

Les Modèles de Markov Cachés (MMC) sont des modèles stochastiques qui permettent de décrire ou d'analyser la génération de séquences discrètes. La théorie des MMC a été élaborée vers la fin des années soixante par Baum et al. [BaP66, BaE67]. Ces modèles ont été très étudiés pour la reconnaissance de la parole depuis le début des années soixante-dix [BAK75]. Mais ce n'est que plus récemment qu'ils ont été employés pour l'analyse de la séquence dans des domaines comme la biologie, la reconnaissance de l'écriture manuscrite ou l'analyse de séquences textuelles.

Un MMC est défini à partir d'une chaîne de Markov et de densités de probabilités. Une chaîne de Markov est un automate à états et est définie d'une part par un ensemble d'états  $S = \{S_1, S_2, \dots, S_l\}$  et d'autre part par une matrice de transitions spécifiant avec quelle probabilité on peut passer d'un état à un autre. Une probabilité de transition entre deux états égale à zéro signifie que cette transition n'est pas autorisée. En spécifiant l'ensemble des transitions autorisées, on obtient différentes topologies de MMC. Les deux structures de MMC les plus répandues sont les MMC ergodiques et les MMC du type Bakis [BKS76]. Dans le cas ergodique tous les états du modèle sont reliés entre eux. La figure 8 (a) montre un MMC ergodique à quatre états. Pour certaines applications, notamment le traitement de signaux temporels comme la parole ou l'écriture, d'autres types de MMC, comme les MMC gauche-droite du type Bakis (figure 8 b) se sont montrés plus efficaces. La propriété fondamentale des MMC gauche-droite est qu'aucune transition n'est autorisée sur les états dont l'indice d'état est inférieur à celui de l'état courant. Avec ces modèles il y a donc toujours un état initial et un état final. A partir de ces deux architectures, il existe beaucoup de combinaisons possibles. Par exemple la figure 8 (c) montre une connexion croisée entre deux MMC parallèles gauche-droite.

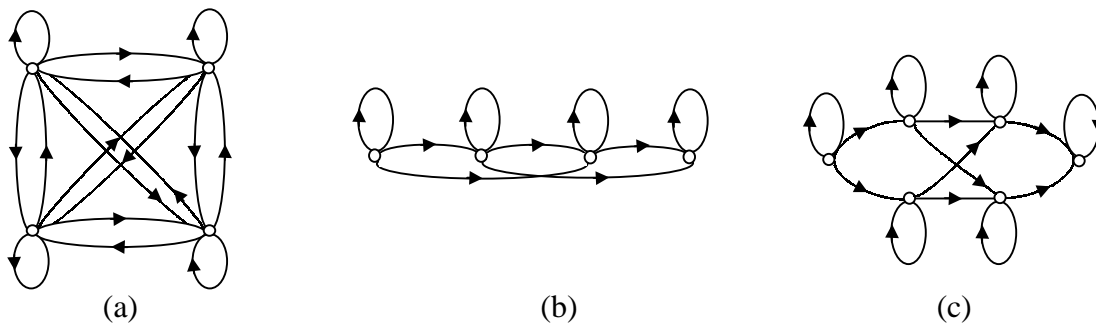


Figure 8. Les trois types de MMC les plus répandues. a) un MMC ergodique à 4 états. b) un MMC gauche-droite à 4 états. c) un MMC parallèle gauche-droite à 6 états.

Dans un MMC, des densités de probabilités, définies sur un espace d'observation, sont associées à chaque état. Un MMC est donc une association d'une chaîne de Markov et de densités de probabilités associées aux états de cette chaîne. Un MMC est un modèle de génération aléatoire de séquences dans l'espace d'observation. Pour générer une séquence avec un MMC, on choisit un état initial, puis on tire au hasard une observation, en utilisant la densité de probabilité associée à l'état courant, puis on tire au hasard l'état suivant, en utilisant la distribution des probabilités de transitions, puis on tire au hasard une nouvelle observation à l'aide de la densité de probabilité associée à l'état courant etc.

Les MMC sont basés sur certaines hypothèses simplificatrices, qui permettent notamment de dériver des algorithmes d'apprentissage très puissants. En considérant  $x = (x_1, x_2, \dots, x_n)$  et  $Q = (q_1, q_2, \dots, q_n)$  respectivement comme une séquence d'observation et une séquence d'états de longueur  $n$ , les deux hypothèses essentielles des MMC sont :

- Etant donné un modèle  $\lambda$ , les observations sont indépendantes :

$$p(x_i / x_1^{i-1}, Q, \lambda) = p(x_i / q_i, \lambda)$$

- la séquence est décrite par une chaîne de Markov d'ordre  $k$  :

$$p(q_i / q_1^{i-1}) = p(q_i / q_{i-1}, \dots, q_{i-k}).$$

Dans ce qui suit, nous allons nous intéresser aux trois problèmes fondamentaux que l'on distingue habituellement dans les MMC [FER80, RaJ93] : l'évaluation de la probabilité (ou la vraisemblance) d'une séquence d'observations étant donné un MMC, la détermination de la meilleure séquence d'états d'un modèle (décodage), l'apprentissage des paramètres du modèle à partir d'un signal observé (apprentissage). La présentation que nous donnons s'inspire largement de [RaJ93].

### 1.3.2 Probabilité d'une séquence d'observation

Nous souhaitons calculer la probabilité d'une séquence d'observation  $x$  étant donné un modèle  $\lambda$ ,  $p(x/\lambda)$ . Cette probabilité peut se calculer en sommant la probabilité jointe  $p(x, Q/\lambda)$  pour toutes les séquences d'états  $Q$  possibles.

$$\begin{aligned} p(x/\lambda) &= \sum_Q p(x, Q/\lambda) = \sum_Q p(x/Q, \lambda).p(Q/\lambda) \\ &= \sum_Q p(q_1).p(x_1/q_1, \lambda) \prod_{i=2}^n p(x_i/x_1^{i-1}, Q, \lambda).p(q_i/q_1^{i-1}) \end{aligned} \quad (1.5)$$

En pratique on utilise des chaînes de Markov d'ordre 1, soit :

$$p(x/\lambda) = \sum_{q_1, q_2, \dots, q_n} p(q_1) \times p(x_1/q_1) \dots p(q_n/q_{n-1}) \times p(x_n/q_n) \quad (1.6)$$

La sommation (1.6) peut être calculée efficacement à l'aide d'une procédure de programmation dynamique connue sous le nom d'algorithme *forward* [BaS68]. Considérons la probabilité  $\alpha_\tau(i)$  d'observer la séquence d'observation partielle  $x_1 x_2 \dots x_\tau$  et d'être dans l'état  $S_i$  au temps  $\tau$  :

$$\alpha_\tau(i) = p(x_1 x_2 \dots x_\tau, q_\tau = S_i / \lambda)$$

On peut calculer les  $\alpha_\tau(i)$  récursivement (algorithme 2).

1) Initialisation

$$\alpha_1(i) = p(q_1 = S_i) \times p(x_1/q_1 = S_i), \quad 1 \leq i \leq l.$$

2) Récursion

$$\alpha_{\tau+1}(j) = \left[ \sum_{i=1}^p \alpha_\tau(i) \times p(q_{\tau+1} = S_j / q_\tau = S_i) \right] \times p(x_{\tau+1} / q_{\tau+1} = S_j), \quad 1 \leq \tau \leq n-1, 1 \leq j \leq l$$

3) Terminaison

$$p(x/\lambda) = \sum_{i=1}^p \alpha_n(i)$$

Algorithme 2. Algorithme *forward* pour le calcul de  $p(x/\lambda)$ .

La figure 9 illustre sous forme de treillis le calcul des probabilités  $\alpha_\tau$  :

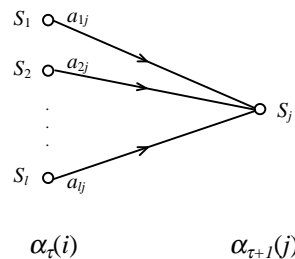


Figure 9. Illustration du mécanisme de calcul de la probabilité de la séquence d'observation par l'algorithme *forward*. Le calcul des  $\alpha_{\tau+1}(j)$  nécessite uniquement de connaître les  $\alpha_\tau(i)$  et les  $a_{ij}$ ,  $i=1, \dots, l$ , où les connexions  $a_{ij}$  représentent les probabilités de transition entre états.

### 1.3.3 Détermination de la séquence d'états optimale

On cherche ici à trouver la meilleure séquence d'états, c'est-à-dire celle qui maximise  $p(Q/x, \lambda)$  ou d'une manière équivalente celle qui maximise  $p(Q, x/\lambda)$ . Pour cela on utilise l'algorithme classique de Viterbi [VIT67, FOR73] qui, comme l'algorithme *forward*, est basé sur des techniques de programmation dynamique.

Nous cherchons à trouver la meilleure séquence d'états  $Q = (q_1, q_2, \dots, q_n)$  pour une séquence d'observation  $x = (x_1, x_2, \dots, x_n)$  donnée. Posons :

$$\delta_\tau(i) = \max_{q_1, q_2, \dots, q_{\tau-1}} p(q_1 q_2 \dots q_\tau = S_i, x_1 x_2 \dots x_\tau / \lambda)$$

$\delta_\tau(i)$  est la probabilité maximale sur un chemin  $x_1, x_2, \dots, x_\tau$  au temps  $\tau$  se terminant à l'état  $S_i$ . De même que dans l'algorithme *forward*, nous obtenons par récurrence :

$$\delta_{\tau+1}(j) = \max_i [\delta_\tau(i) \cdot p(q_{\tau+1} = S_j / q_\tau = S_i)] \times p(x_{\tau+1} / q_{\tau+1} = S_j) \quad (1.7)$$

Pour trouver la séquence d'états optimale, nous allons introduire un tableau  $\psi_\tau(j)$  pour mémoriser l'argument à maximiser dans (1.7). La procédure complète pour trouver la suite d'états optimale est alors la suivante, (algorithme 3).

---

1) Initialisation

$$\delta_1(i) = p(q_1 = S_i) \times p(x_1 / q_1 = S_i), \quad 1 \leq i \leq l.$$

$$\psi_1(i) = 0.$$

2) Récursion

$$\delta_{\tau+1}(j) = \arg \max_{1 \leq i \leq l} [\delta_\tau(i) \times p(q_{\tau+1} = S_j / q_\tau = S_i)] \times p(x_{\tau+1} / q_{\tau+1} = S_j), \quad 1 \leq \tau \leq n-1, 1 \leq j \leq l$$

$$\psi_{\tau+1}(j) = \arg \max_{1 \leq i \leq l} [\delta_\tau(i) \times p(q_{\tau+1} = S_j / q_\tau = S_i)], \quad 1 \leq \tau \leq n-1, 1 \leq j \leq l$$

3) Terminaison

$$q_n^* = \arg \max_{1 \leq i \leq l} [\delta_n(i)]$$

La séquence d'états optimale est alors obtenue par « *backtracking* » :

$$q_\tau^* = \psi_{\tau+1}(q_{\tau+1}^*), \quad \tau = n-1, n-2, \dots, 1.$$


---

Algorithme 3. Algorithme de *Viterbi* pour trouver la séquence d'états optimale.

Il est à noter que l'algorithme de Viterbi est similaire à l'algorithme *forward*. La différence majeure réside dans le fait qu'au lieu de sommer à l'étape *Récursion* dans l'algorithme de *forward* on maximise dans l'algorithme de Viterbi.

### 1.3.4 Apprentissage des paramètres

Le troisième problème des MMC est d'apprendre les paramètres du modèle (les probabilités de transition et d'émission) de façon à maximiser la probabilité de la séquence d'observation étant donné le modèle, i.e.  $p(x/\lambda)$ . Il existe différentes méthodes itératives pour maximiser localement cette probabilité, comme l'algorithme de Baum-Welch qui est une instance de l'algorithme EM<sup>7</sup> [DEM77] ou comme les techniques de

---

<sup>7</sup> Nous allons revenir en détail sur cet algorithme au chapitre 2, section 2.3.3.1



gradient [LEV83]. Nous allons présenter l'algorithme développé par Baum et al. qui est le plus communément utilisé. Cette méthode utilise les variables  $\alpha_\tau$  calculées par l'algorithme *forward* et des variables  $\beta_\tau$  calculées par l'algorithme *backward*. Ce dernier algorithme est similaire à l'algorithme *forward* mais dans le sens inverse, i.e. nous souhaitons calculer la probabilité de la séquence d'observations partielle du temps  $\tau+1$  jusqu'à la fin étant donné l'état  $S_i$  au temps  $\tau$  et le modèle  $\lambda$ . En notant  $\beta_\tau(i) = p(x_{\tau+1} x_{\tau+2} \dots x_n / q_\tau = S_i, \lambda)$ , ces probabilités  $\beta_\tau(i)$  s'obtiennent itérativement comme ci-dessous, (algorithme 4).

1) Initialisation

$$\beta_n(i) = 1, 1 \leq i \leq l$$

2) Récursion

$$\beta_\tau(i) = \sum_{j=1}^l p(q_{\tau+1} = S_j / q_\tau = S_i) \times p(x_{\tau+1} / q_{\tau+1} = S_j) \times \beta_{\tau+1}(j), \tau = n-1, \dots, 1, 1 \leq i \leq l$$

Algorithme 4. Algorithme *backward* pour le calcul des  $\beta$

L'étape d'initialisation définit d'une manière arbitraire les  $\beta_n(i)=1$  pour tous les états  $i$ . La figure 10 illustre l'étape de récursion. Pour être dans l'état  $S_i$  au temps  $\tau$  et avoir la séquence d'observation partielle du temps  $\tau+1$ , il faut considérer tous les états possibles  $S_j$  au temps  $\tau+1$ , ainsi que les probabilités de transition de l'état  $S_i$  à l'état  $S_j$  et les probabilités d'émission des observations partielles dans les états  $S_j$  ainsi que les  $\beta_{\tau+1}(j)$ .

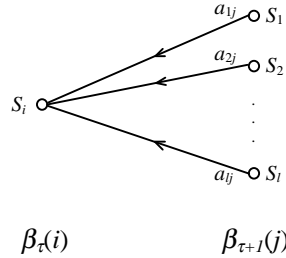


Figure 10. Illustration du mécanisme de calcul de la probabilité de la séquence d'observation par l'algorithme *backward*.

Nous introduisons maintenant d'autres variables  $\xi_\tau(i,j)$  et  $\gamma_\tau(i)$  qui vont nous permettre d'écrire plus simplement l'algorithme. Posons  $\xi_\tau(i,j) = p(q_\tau = S_i, q_{\tau+1} = S_j / x, \lambda)$ , la probabilité d'être dans l'état  $S_i$  au temps  $\tau$  et  $S_j$  au temps  $\tau+1$  étant donné le modèle et la séquence d'observation. En utilisant les variables *forward* et *backward* nous pouvons écrire,

$$\begin{aligned} \xi_\tau(i, j) &= \frac{p(q_\tau = S_i, q_{\tau+1} = S_j, x / \lambda)}{P(t / \lambda)} \\ &= \frac{\alpha_\tau(i) \cdot a_{ij} \cdot b_j(x_{\tau+1}) \cdot \beta_{\tau+1}(j)}{\sum_{i=1}^l \sum_{j=1}^l \alpha_\tau(i) \cdot a_{ij} \cdot b_j(x_{\tau+1}) \cdot \beta_{\tau+1}(j)} \end{aligned}$$

où  $a_{ij}$  est la probabilité de transition de l'état  $S_i$  à l'état  $S_j$  et  $b_j(x_{\tau+1})$  est la probabilité d'émission de l'observation  $x_{\tau+1}$  par l'état  $S_j$ . Notons maintenant  $\gamma_\tau(i)$  la probabilité d'être dans l'état  $S_i$  au temps  $\tau$ ;  $\gamma_\tau(i)$  peut être obtenu en sommant  $\xi_\tau(i, j)$  suivant  $j$  :

$$\gamma_\tau(i) = \sum_{j=1}^l \xi_\tau(i, j)$$

Les sommations de  $\xi_\tau(i, j)$  et  $\gamma_\tau(i)$  suivant  $\tau$ , i.e.  $\sum_{\tau=1}^{n-1} \xi_\tau(i, j)$  et  $\sum_{\tau=1}^{n-1} \gamma_\tau(i)$ , peuvent être interprétées respectivement comme le nombre de transitions entre l'état  $S_i$  et l'état  $S_j$ , et le nombre total de transitions à partir de l'état  $S_i$ . Les paramètres du MMC peuvent être estimés en utilisant ces deux sommes. La distribution initiale de probabilités des états :

$$\bar{p}(x_1 = S_i) = \text{La fréquence d'apparition dans l'état } S_i \text{ au temps } (\tau = 1) = \gamma_1(i)$$

Les probabilités de transition entre les états  $S_i$  et  $S_j$  :

$$\bar{p}(q_{\tau+1} = S_j / q_\tau = S_i) = \frac{\text{Le nombre de transitions entre les états } S_i \text{ et } S_j}{\text{Le nombre total de transitions de l'état } S_i} = \frac{\sum_{\tau=1}^{n-1} \xi_\tau(i, j)}{\sum_{\tau=1}^{n-1} \gamma_\tau(i)}$$

Les probabilités d'émissions de l'état  $S_j$  observant  $v_k$  :

$$\bar{p}(v_k \text{ au temps } \tau / q_\tau = S_j) = \frac{\text{Le nombre de fois où on observe } v_k \text{ dans l'état } S_j}{\text{Le nombre de fois où l'on est dans l'état } S_j} = \frac{\sum_{\tau=1}^n \gamma_\tau(j)}{\sum_{\tau=1}^n \gamma_\tau(j)}$$

Si on utilise d'une manière itérative cette procédure d'estimation des paramètres, la probabilité d'observation de la séquence  $x$  augmente jusqu'à ce qu'un maximum local de la vraisemblance soit atteint (cf. chapitre 2).

## 1.4 Machines à Vecteurs Support

### 1.4.1 Introduction

Pour une tâche d'apprentissage donnée, avec un nombre fini de données d'apprentissage, la meilleure performance en généralisation (c'est-à-dire sur des données non observées pendant l'apprentissage) est obtenue si on arrive à faire un bon compromis entre la nature des données et la capacité du système choisi à apprendre ces données. Un système avec une grande capacité est comme un bachelier qui apprend par cœur mais qui se trompe souvent sur des exemples non appris et un système avec une faible capacité est comme son frère qui n'apprend pas du tout et qui est incapable de bien généraliser.

L'exploration et la formulation de ces concepts ont donné naissance à une théorie dans le domaine de l'apprentissage statistique [VAP82], d'où sont issues les Machines à Vecteurs Support (MVS). Dans ce qui suit nous allons présenter les MVS, développées par Vapnik, pour une tâche de classification et pour un problème à deux classes. Nous

commençons par décrire les MVS linéaires au paragraphe 1.4.2, puis nous présenterons les MVS non linéaires au paragraphe 1.4.3.

## 1.4.2 MVS Linéaires

Dans ce paragraphe, nous allons présenter les modèles MVS linéaires en distinguant deux cas, le cas de données linéairement séparables, et le cas de données non linéairement séparables.

### 1.4.2.1 Cas séparable

Nous envisageons dans un premier temps le cas de données linéairement séparables. Le cas général des données non linéairement séparables sera présenté comme une extension au paragraphe suivant.

Soit  $\{x_i, t_i\}, i=1, \dots, n$ , l'ensemble des données d'apprentissage  $x_i \in \mathbb{R}^p$  avec leur étiquette  $t_i \in \{-1, 1\}$ . Supposons qu'il existe un hyperplan qui sépare les exemples positifs ( $t_i = 1$ ) des exemples négatifs ( $t_i = -1$ ). L'équation de l'hyperplan est  $w \cdot x + b = 0$ , où  $w$  représente le vecteur normal à l'hyperplan,  $|b|/\|w\|$  la distance de l'hyperplan à l'origine et  $\|w\|$  la norme euclidienne de  $w$ .

Posons  $d_+$  (respectivement  $d_-$ ) la distance entre le plus proche point de la classe des exemples positifs (respectivement négatifs) et l'hyperplan. On définit la *marge* d'un hyperplan comme la somme  $d_+ + d_-$  (figure 11).

Pour l'hyperplan, tous les exemples de la base d'apprentissage vérifient les deux contraintes suivantes :

$$x_i \cdot w + b \geq 0 \text{ pour } t_i = +1$$

$$x_i \cdot w + b \leq 0 \text{ pour } t_i = -1$$

Moyennant une normalisation des paramètres on peut réécrire ces équations sous la forme, qui sera celle utilisée par la suite :

$$x_i \cdot w + b \geq +1 \text{ pour } t_i = +1 \tag{1.8}$$

$$x_i \cdot w + b \leq -1 \text{ pour } t_i = -1 \tag{1.9}$$

Ces deux conditions peuvent se combiner en un seul ensemble d'inéquations :

$$\forall i, t_i(x_i \cdot w + b) - 1 \geq 0 \tag{1.10}$$

Supposons qu'il existe des points qui vérifient l'égalité dans l'inéquation (1.8). Ces points appartiennent à l'hyperplan  $H_1 : x_i \cdot w + b = 1$ . De même, considérons les points vérifiant l'égalité dans l'inéquation (1.9), ces points appartiennent à l'hyperplan  $H_2 : x_i \cdot w + b = -1$ . Dans ces conditions,  $d_+ = d_- = 1/\|w\|$  et la marge vaut  $2/\|w\|$ .  $H_1$  et  $H_2$  sont parallèles et distincts. En minimisant  $\|w\|^2$  sous les contraintes (1.10), on peut donc trouver une paire d'hyperplans qui maximise la marge. La figure 11 illustre ces faits dans le cas d'un problème à deux dimensions. Parmi les points d'apprentissage, ceux qui vérifient l'égalité de l'inéquation (1.10), (les points qui appartiennent aux hyperplans  $H_1$  et  $H_2$ ) sont appelés les vecteurs de support (sur la figure 11 ils sont indiqués par un rond les encerclant). Pour résoudre ce problème de minimisation sous contraintes, nous allons utiliser la formulation lagrangienne du problème. Les raisons en sont :

- que les points de l'ensemble d'apprentissage apparaissent sous la forme d'un produit scalaire,
- et que les variables lagrangiennes sont plus aisées à manipuler que les  $N$  inéquations de (1.10).

La formulation lagrangienne d'un problème du type  $c_i \geq 0$  consiste à multiplier les inéquations par des multiplicateurs de Lagrange positifs et à les soustraire de la fonction objective (i.e. la fonction à optimiser).

Dans notre cas, la fonction objective est la norme des poids  $w$  et les contraintes sont les inéquations (1.10), ce qui donne le *lagrangien primal*  $L_P$  :

$$L_P = \frac{1}{2} \|w\|^2 - \sum_{i=1}^N \alpha_i [t_i (x_i \cdot w + b) - 1] \quad (1.11)$$

Où les  $\alpha_i$  sont les multiplicateurs de Lagrange. Pour ce problème primal, il faut minimiser  $L_P$  en fonction de  $w$  et  $b$  et simultanément avoir des dérivées nulles de  $L_P$  par rapport aux  $\alpha_i$  (appelons  $C_1$  cet ensemble de contraintes). Comme la fonction objective est convexe et que les points qui satisfont les contraintes forment aussi un ensemble convexe, la résolution de ce problème relève d'un problème de programmation quadratique.

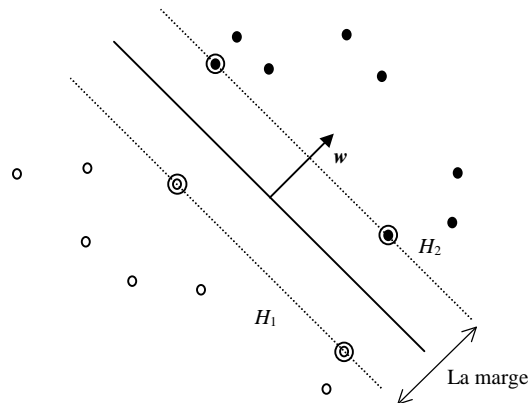


Figure 11. Les hyperplans pour un problème de classification linéairement séparable. Les points à support vecteurs sont encadrés.

D'une manière équivalente nous pouvons résoudre le problème dual suivant : maximiser  $L_P$ , en annulant le gradient de  $L_P$  en fonction de  $w$  et de  $b$ , avec des  $\alpha_i$  positifs (appelons  $C_2$  cet ensemble de contraintes). Cette formulation duale du problème est appelée le dual de Wolfe [FLE87] et possède la propriété que le maximum de  $L_P$  relatif aux contraintes  $C_2$  se manifeste aux mêmes valeurs de  $w$ ,  $b$  et  $\alpha$  que le minimum de  $L_P$  relatif aux contraintes  $C_1$ . L'annulation du gradient par rapport à  $w$  et à  $b$  donne :

$$\nabla_w L_P = w - \sum_{i=1}^N \alpha_i t_i x_i = 0,$$

$$\frac{\partial L_P}{\partial b} = \sum_{i=1}^N \alpha_i t_i = 0$$

Soit :

$$w = \sum_{i=1}^N \alpha_i t_i x_i,$$

$$\sum_{i=1}^N \alpha_i t_i = 0$$

En substituant ces égalités dans le primal (1.11) on obtient :

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i H_{ij} \alpha_j \quad (1.12)$$

où  $H_{ij} = t_i t_j x_i \cdot x_j$ .

Pour le cas linéaire, les vecteurs support sont obtenus en maximisant  $L_D$  en fonction des  $\alpha_i$ . Dans la solution, à chaque point est associé un multiplicateur de Lagrange  $\alpha_i$ . Les points pour lesquels  $\alpha_i > 0$  sont appelés «Vecteur Support» et appartiennent à  $H_1$  ou  $H_2$ . Tous les autres exemples d'apprentissage pour lesquels  $\alpha_i = 0$  sont de part et d'autre de ces hyperplans. Ces derniers points n'influent pas sur la frontière de décision.

#### 1.4.2.2 Cas non-séparable

L'algorithme ci-dessus n'a pas de solution exacte dans le cas non-séparable. Dans ce cas, Vapnik propose d'introduire de nouvelles variables  $\xi_i$ , pour assouplir les contraintes dans les inéquations (1.8) et (1.9), [COR95], i.e.

$$x_i \cdot w + b \geq +1 - \xi_i \text{ pour } t_i = +1 \quad (1.13)$$

$$x_i \cdot w + b \leq -1 + \xi_i \text{ pour } t_i = -1 \quad (1.14)$$

$$\forall i, \xi_i \geq 0 \quad (1.15)$$

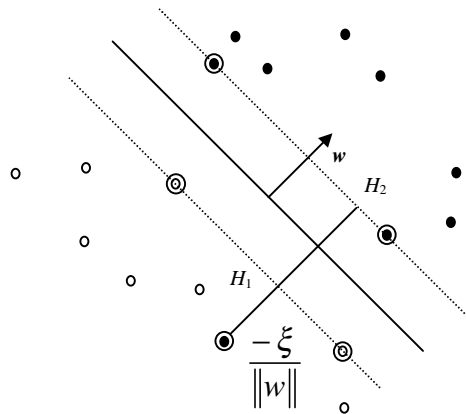


Figure 12. Les hyperplans linéaires pour un problème de classification non linéairement séparable. Lorsqu'il y a une erreur sur un exemple, cet exemple est considéré comme un vecteur support dont sa distance avec l'hyperplan de sa vraie classe est  $-\xi/\|w\|$ .

D'après (1.13) et (1.14) lorsqu'il y a une erreur pour un point  $x_i$  donné, la variable  $\xi_i$  correspondante doit être plus grande que l'unité, puisque dans ce cas  $+1-\xi_i$  ou  $-1+\xi_i$

changent de signe. Par ce fait  $\sum_i \xi_i$  détermine une borne supérieure du nombre d'erreurs en apprentissage. Ainsi, une façon simple de traiter ce cas est de changer la fonction objective du problème primal de  $\|w\|^2/2$  en  $\|w\|^2/2 + C$ , où  $C$  est un paramètre fixé à l'avance. Plus  $C$  est grand plus on pénalise les erreurs. La figure 12 illustre ce cas. Comme précédemment le lagrangien de ce problème s'écrit :

$$L_p = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i \{t_i(x_i \cdot w + b) - 1 + \xi_i\} - \sum_{i=1}^N \mu_i \xi_i \quad (1.16)$$

Où les  $\mu_i$  sont des multiplicateurs de Lagrange introduits pour prendre en compte les contraintes de positivité des  $\xi_i$ . Les vecteurs support sont obtenus en résolvant les inéquations suivantes :

$$\begin{aligned} \frac{\partial L_p}{\partial w_v} &= w_v - \sum_{i=1}^N \alpha_i t_i x_{iv} \\ \frac{\partial L_p}{\partial b} &= - \sum_{i=1}^N \alpha_i t_i \\ \frac{\partial L_p}{\partial \xi_i} &= C - \alpha_i - \mu_i \\ t_i \cdot (x_i \cdot w + b) - 1 + \xi_i &\geq 0 \\ \xi_i \geq 0, \alpha_i \geq 0, \mu_i &\geq 0. \end{aligned}$$

### 1.4.3 MVS Non linéaires

[BOS92] ont montré que les méthodes présentées ci-dessus pouvaient être généralisées dans le cas où les fonctions de décision ne seraient plus des fonctions linéaires en  $x$ . En se basant sur les travaux de Aizerman [AIZ64], ils ont établi que dans, certains cas, en plongeant les données non linéairement séparables dans un espace euclidien de plus grande dimension (voire infinie), les données peuvent devenir linéairement séparables<sup>8</sup> dans ce nouvel espace, ce qui permet de se ramener au cas linéairement séparable présenté dans le paragraphe précédent, figure 13.

Nous allons définir brièvement les fonctions noyaux et leur utilité pour obtenir des vecteurs support dans le cas où nous plongeons les données dans un espace de plus grande dimension appelé aussi *espace des caractéristiques*.

Notons d'abord que dans l'équation (1.12) les données n'apparaissent que sous la forme de produits scalaires  $x_i \cdot x_j$ . Notons  $\Phi : X \rightarrow \Xi$  la fonction qui plonge les données de  $X$  dans un espace euclidien  $\Xi$ , où  $\dim \Xi > \dim X$ . De ce fait, ce sont des termes de la forme  $\Phi(x_i) \cdot \Phi(x_j)$  qui apparaissent dans l'équation (1.12). Supposons maintenant qu'il existe une fonction noyau  $K$  tel que  $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$ . On peut alors utiliser  $K$  dans cette formulation sans connaître explicitement l'expression de  $\Phi$ .

<sup>8</sup> Le théorème de Mercer est à la base de la faisabilité de cette procédure. Il permet d'interpréter les noyaux comme des produits scalaires dans l'espace des caractéristiques. Ce théorème a été introduit par [AIZ64] dans le domaine de l'apprentissage mais il date de 1909 [MER09].

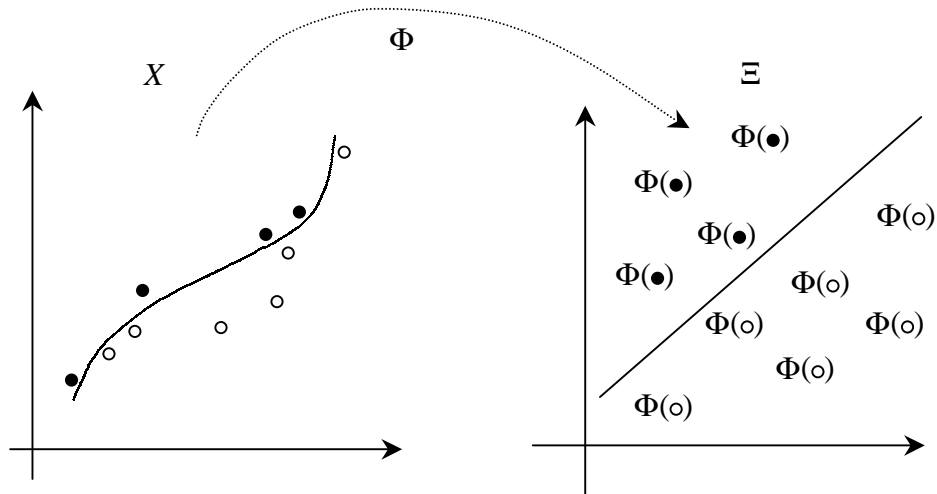


Figure 13. En plongeant les données dans un espace de plus grande dimension, on peut simplifier la tâche de classification.

Le résultat intéressant de cette hypothèse est qu'en remplaçant  $x_i \cdot x_j$  par  $K(x_i, x_j)$  dans l'algorithme d'apprentissage, même si l'espace des caractéristiques est de dimension infini, l'algorithme produit des vecteurs support aussi rapidement que pour un problème dans un espace de dimension finie. Dans la phase de test, la MVS est utilisée pour calculer le signe de la fonction :

$$f(x) = \sum_{j=1}^{N_s} \alpha_j t_j K(s_j, x) + b \quad (1.17)$$

où  $s_j, j=\{1, \dots, N_s\}$  sont les vecteurs support. Les premières fonctions noyaux étudiées dans le domaine de la reconnaissance des formes étaient :

$$K(x, y) = (x \cdot y + 1)^p \quad (1.18)$$

$$K(x, y) = e^{-\|x-y\|^2 / 2\sigma^2} \quad (1.19)$$

$$K(x, y) = \tanh(\kappa x \cdot y - \delta) \quad (1.20)$$

L'équation (1.18) correspond à un classifieur polynomial de degré  $p$ , l'équation (1.19) à un classifieur basé sur des gaussiennes à base radiale (FBR) et l'équation (1.20) à une forme particulière de réseau de neurones avec fonctions d'activation sigmoïdales.

Dans le cas des FBR, le nombre de centres ( $N_s$  dans l'équation (1.17)), les centres eux-mêmes  $s_j$ , les poids  $\alpha_j$  et le seuil  $b$  sont déterminés automatiquement dans la phase d'apprentissage de la MVS. Pour le cas des réseaux de neurones, la première couche est constituée de  $N_s$  ensembles de poids, où chaque ensemble est constitué de  $d_p$  (la dimension des entrées) poids. Et la seconde est constituée de  $N_s$  poids (les  $\alpha_j$ ), l'évaluation se fait alors en prenant une somme pondérée de sigmoïdes, qui sont calculées d'après (1.20) par un produit scalaire des exemples de test avec les vecteurs support. L'architecture des réseaux (le nombre de poids) est déterminée par la MVS en apprentissage.

Les classifieurs MVS décrits ici sont des classifieurs binaires, mais ils sont très facilement transposables pour traiter le cas multi-classes. Le cas à  $C$  classes est habituellement traité comme  $C$  classifieurs à 2 classes : pour  $i = 1, \dots, C$ , on prend la classe  $i$  contre les autres classes. La décision est d'attribuer un point à la classe de plus fort score. D'autres traitements du cas multi-classes ont été proposés.

Les classifieurs MVS sont une nouvelle famille de classifieurs non linéaires au même titre que les réseaux de neurones, les arbres ou les nombreuses autres méthodes proposées dans les années 90. Ils présentent l'avantage d'être décrits par une formulation statistique qui intègre plusieurs idées développées dans le courant de ces années. Néanmoins, en pratique, ils demandent comme les autres systèmes non linéaires le réglage semi-manuel de différents paramètres. Les performances de ces différentes classes de machines sont en général équivalentes.

### 1.5 Conclusion

Dans ce chapitre nous avons étudié quelques modèles de l'apprentissage numérique, les réseaux de neurones, les MMC et les MVS, modèles que nous avons utilisés dans nos expériences présentées dans la partie 3. Les PMC et les MMC ont été employés pour modéliser les séquences de termes (voir chapitre 6). Et dans le chapitre 7 nous avons utilisé les PMC et les MVS pour faire de la classification de phrases.



## 1.6 Bibliographie

- [ABD88] H. Abdi. Generalized approaches for connectionist auto-associative memories: Interpretation, implication and illustration for face processing. *Artificial intelligence and cognitive sciences*. Manchester University Press, 1988.
- [AIZ64] M. A. Aizerman, E. M. Braverman, L. I. Rozoner. Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning. *Automation and Remote Control*, Vol. 25, pp. 821--837, 1964.
- [AnR88] J. A. Anderson, E. Rosenfeld. Neurocomputing: Foundations of Research. *MIT Press*, 1988.
- [BAK75] J. A. Baker. The Dragon system – An overview. *IEEE Transactions on Acoustic Speech Signal Processing*, Vol. ASSP-23, N°1, pp.24--29, 1975.
- [BKS76] R. Bakis. Continuous speech recognition via centi-second acoustic states. *In Proceedings ASA Meeting*. 1976.
- [BaP66] L. E. Baum, T. Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *Annual Mathematical Statistics*, vol. 37, pp.1554--1563, 1966.
- [BaE67] L. E. Baum, J. A. Egon. An inequality with Applications to Statistical Estimation for Probabilistic Functions of a Markov process and to a Model for Ecology. *Bull. Amer. Meteorol. Soc.*, vol. 73, pp. 360--363, 1967.
- [BaS68] L. E. Baum, G. R. Sell. Growth Functions for Transformations on Manifolds. *Pac. J. Math.*, vol. 27, N°2, pp. 211--227, 1968.
- [BIS95] C. M. Bishop. Neural Networks for Pattern Recognition. *Clarendon Press*, Oxford, 1995.
- [BLO62] H. D. Block. The Perceptron: a model for brain functioning. *Reviews of Modern Physics*, vol. 34, N°1, pp. 123--135.
- [BOS92] B. E. Boser, I. M. Guyon, V. Vapnik. A Training Algorithm for Optimal Margin Classifier. *In Fifth Annual Workshop on Computational Learning Theory*. Pittsburgh, 1992.
- [CHA83] J. P. Changeux. L'homme neuronal. *Fayard*, 1983.
- [COR95] C. Cortes, V. Vapnik. Support Vector Networks, *Machine Learning*, Vol. 20, pp. 273--297, 1995.
- [DEM77] A. P. Dempster, N. M. Laird, D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*. Vol. 39, N° 1, pp.1--38, 1977.

- [DuH73] R. O. Duda, P. E. Hart. Pattern Classification and Scene Analysis. *John Wiley*, New York, 1973.
- [FER80] J.D. Ferguson. Variable duration models for speech. *Proc. Symposium on the Application of Hidden Markov Models to Text and Speech*, pp. 143--179, Princeton, NJ, 1980.
- [FLE87] R. Fletcher. Practical Methods of Optimization. *John Wiley et Sons, Inc.*, 2<sup>nd</sup> edition, 1987.
- [FOR73] G. D. Forney. The Viterbi algorithm. *Proc. IEEE*, Vol N° 61, pp. 268--278, 1973.
- [HEB49] D. Hebb. The Organization of Behavior. *John Wiley*, New York, 1949.
- [HOP84] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences USA*, vol. 81, pp. 3088--3092, Mai 1984.
- [LEV83] S. E. Levinson, L. R. Rabiner, M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of a Markov process to Automatic Speech recognition. *Bell Systems Technical Journal*, Vol. 62, N° 4, pp. 1035--1074, 1983.
- [MER09] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophy Transactions by Royal Society of London*. Vol. A, N° 209, pp.415--446, 1909.
- [McP43] W. S. McCulloch, W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin de Mathematical Biophysics*, N° 5 pp. 115--133, 1943.
- [MIN52] M. Minsky. A Neural-Analogue Calculator Based upon a Probability Model of Reinforcement. *Harvard University Psychological Laboratories, Cambridge, Massachusetts*, 8 Janvier 1952.
- [MiP69] M. Minsky, S. Papert. Perceptrons: An Introduction to Computational Geometry. *The MIT Press*, 1969.
- [NIL65] N.J. Nilsson. Learning machines. *McGraw-Hill, New York*, 1965.
- [RaJ93] L. Rabiner, B.-H. Juang. Fundamentals of Speech Recognition. *Prentice Hall*, 1993.
- [REN89] S. Renals, R. Rohwer. Phoneme Classification Experiments Using Radial Basis Functions. *International Joint Conference on Neural Networks*, Vol. 1, pp. 461--467, Washington, D.C., 1989.

- [ROS58] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, N° 65, pp. 386--408.
- [RuM86] D. E. Rumelhart, J. L. McClelland. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, *volume 1-3*. MIT Press, 1986.
- [TUR50] A. M. Turing. Computing machinery and intelligence. *Mind*, LIX(236) pp.433--460, 1950.
- [VAP82] V. N. Vapnik. Learning Dependencies based on Empirical Data. Springer, New York, 1982.
- [VIT67] A. J. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically optimal decoding Algorithm. *IEEE Transactions on Information Theory*, Vol. IT-13, pp.260--269, 1967.
- [WiH60] G. Widrow, M. Hoff. Adaptive switching circuits. In *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record*, Part 4, pp. 96--104, 1960.



# Chapitre 2

## Apprentissage supervisé et non-supervisé

**Résumé.** Dans ce chapitre nous allons présenter succinctement les méthodes d'apprentissage supervisé et non-supervisé. Pour les méthodes supervisées, nous allons nous intéresser notamment aux notions fondamentales qui ont suscité beaucoup d'intérêt dans la communauté de l'apprentissage comme l'optimisation, la généralisation et la complexité. Nous illustrerons chacune de ces notions par un problème particulier. Dans la deuxième partie nous présenterons la problématique de l'apprentissage non-supervisé et nous aborderons des problèmes tels que l'estimation d'un mélange de densités, les algorithmes EM et CEM associés à des critères tels que la vraisemblance du mélange et la vraisemblance de classification.

**Mots clés :** Apprentissage supervisé, algorithmes d'optimisation, problème de généralisation, apprentissage non-supervisé, algorithme EM, La vraisemblance de classification, algorithme CEM.

### 2.1 Introduction

Dans ce chapitre nous allons présenter un ensemble de méthodes d'optimisation pour l'apprentissage *supervisé* et l'apprentissage *non-supervisé*. Ces techniques ont été utilisées pour traiter les problèmes auxquels nous nous sommes intéressés. Comme les méthodes utilisées diffèrent suivant qu'il s'agit d'apprentissage supervisé ou non-supervisé, nous garderons cette séparation pour les présenter.

- En apprentissage *supervisé*, on veut apprendre la relation probabiliste (ou la distribution jointe)  $p(x,t)$  entre les exemples (ou formes d'entrée)  $x \in X$  et les formes désirées  $t \in T$ . Cet apprentissage se fait habituellement à l'aide d'exemples étiquetés  $\{(x_i, t_i) / i=1, \dots, n\}$  issus de la probabilité jointe  $p(x,t)$ . Entrent dans ce cadre la discrimination (avec  $T$  fini) et la régression ( $T \subset \mathbb{R}^N$ ).
- En apprentissage *non-supervisé*, on ne possède pas en général assez d'information pour formuler un critère d'apprentissage pertinent. Dans ce cas on essaie de trouver la structure interne des données issues d'une distribution inconnue  $p(x)$  (appelée aussi source) à partir d'un ensemble de données d'apprentissage  $\{x_i / i = 1, \dots, m\}$ . Les algorithmes non-supervisés sont utilisés pour la visualisation (c'est le cas des

analyses factorielles [EVE84] comme l'ACP et l'analyse canonique [BIS95]), la détection de similarité (algorithme de regroupement comme les *k-moyennes* [DuH73]) et l'estimation de densité (modèle de mélange de densités [McL88, RED84, TIT85]).

Ce chapitre est composé de deux parties. Nous allons détailler dans la première partie les techniques d'apprentissage supervisé (section 2.2) et dans la seconde celles d'apprentissage non-supervisé (section 2.3). Nous ne donnerons pas une description exhaustive des algorithmes développés dans ces domaines mais nous aborderons ceux utilisés dans notre travail pour l'accès à l'information textuelle.

## 2.2 Apprentissage supervisé

### 2.2.1 Introduction

En apprentissage supervisé on cherche à optimiser un critère qui mesure la similarité entre les sorties du système appris et les sorties désirées. L'apprentissage fait appel à des algorithmes d'optimisation en continu ou en discret suivant la nature du problème et la technique d'apprentissage utilisée.

A la section 2.2.2 nous présenterons le problème d'optimisation en introduisant la différence entre l'apprentissage en-ligne et l'apprentissage hors-ligne. Nous étudierons ensuite les algorithmes d'optimisation (section 2.2.3) et aborderons la problématique de la généralisation, en illustrant ce problème par le dilemme biais-variance (section 2.2.5), nous terminons cette section en étudiant les algorithmes de Bagging et de Boosting qui sont une voie très explorée depuis quelques années pour améliorer les capacités de généralisation des systèmes d'apprentissage (section 2.2.6)

### 2.2.2 Problème d'optimisation

#### 2.2.2.1 Apprentissage hors-ligne et en-ligne

On distingue deux classes d'algorithmes d'apprentissage suivant qu'ils opèrent hors-ligne ou en-ligne.

- Dans le cas de *l'apprentissage hors-ligne*, on désire apprendre une fois pour toutes les paramètres d'un système. Pour cela on utilise un ensemble d'apprentissage de taille fixe que l'on exploite pour mettre au point la machine d'apprentissage qui est figée par la suite.
- Dans le cas de *l'apprentissage en-ligne*, on apprend en continu pendant l'utilisation du système. C'est le cas, par exemple des algorithmes de traitement adaptatif du signal. L'intérêt de cette démarche repose dans la capacité du système à s'adapter à des variations de l'environnement, comme l'évolution de la nature du bruit sur une ligne téléphonique.

On peut aussi imaginer des systèmes mixtes, initialisés par apprentissage hors-ligne, et capables par la suite d'adaptation. Nous avons par exemple mis au point un Système d'Aide au Résumé Automatique (S.A.R.A.) basé sur cette idée, ce système est étudié en détail dans le chapitre 7.

### 2.2.2.2 Procédure stochastique pour l'adaptation et l'apprentissage

Avec les techniques supervisées, le critère que l'on optimise est souvent un critère global, comme la minimisation d'une probabilité d'erreur, que l'on peut évaluer en observant le comportement du système sur un ensemble de formes représentatif des conditions extérieures.

Les algorithmes stochastiques qui adaptent les paramètres du système d'apprentissage séquentiellement avec l'arrivée des exemples sont fréquemment utilisés pour l'apprentissage en-ligne. Ils ont également été largement utilisés pour l'apprentissage hors-ligne, où ils peuvent se révéler plus rapides que les algorithmes d'optimisation classiques, notamment pour des problèmes de grande dimension ou dans le cas d'exemples redondants [BOT91].

### 2.2.3 Algorithmes d'optimisation

#### 2.2.3.1 Critère global

Posons  $F$  la fonction de transfert du système. Un des critères globaux très utilisé dans la théorie de l'apprentissage supervisé est de la forme [TSY73] :

$$C = E_{x,t}[G(F(x,w),t)] \quad (2.1)$$

On note  $E_{x,t}[\cdot]$  l'espérance mathématique associée aux variables aléatoires  $x$  et  $t$ ,  $G(F(x,w),t)$  représente la mesure, peu fiable, du critère global sur un seul exemple. Par exemple la distance euclidienne entre le désiré  $t$  et la sortie du modèle. Cette formule peut s'écrire sous la forme intégrale en introduisant la densité de probabilité jointe  $p(x,t)$  associée aux variables aléatoires  $x$  et  $t$ .

$$C = \iint G(F(x,w),t).p(x,t) dx dt \quad (2.2)$$

Notons que l'on connaît analytiquement  $F(x,w)$  et  $G(F(x,w),t)$  mais que la densité  $p(x,t)$  est généralement inconnue.

#### 2.2.3.2 Descente de gradient

Dans la technique de descente de gradient on se propose de chercher l'ensemble des paramètres  $w$  minimisant le critère

$$\iint G(F(x,w),t).p(x,t) dx dt \quad (2.3)$$

L'algorithme classique de descente de gradient, appelé aussi gradient déterministe, consiste à itérer :

$$w_\tau = w_{\tau-1} - \eta_\tau \cdot \nabla_w \left( \iint G(F(x,w),t).p(x,t) dx dt \right) \quad (2.4)$$

Où l'opérateur nabla,  $\nabla_w$ , dénote le gradient par rapport aux paramètres  $w$ . Dans cette équation le gain  $\eta_\tau$  peut être soit un réel positif, soit une matrice symétrique positive.

Dans le cas discret, on souhaite optimiser le critère (2.4) à l'aide d'un échantillon d'exemples  $\{x_i, t_i\}$  de la base d'apprentissage, soit :

$$w_\tau = w_{\tau-1} - \eta_\tau \nabla C(w_\tau) = w_{\tau-1} - \frac{\eta_\tau}{n} \sum_{i=1}^n \nabla G(F(x_i, w_{\tau-1}), t_i) \quad (2.5)$$

D'après la loi des grands nombres on sait que la moyenne empirique des gradients de  $G$  converge vers son espérance mathématique. De plus, si  $G$  est différentiable et que son gradient est intégrable, l'espérance du gradient est égale au gradient de l'espérance :

$$\lim_{n \rightarrow +\infty} \left( \frac{1}{n} \sum_{i=1}^n \nabla G(F(x_i, w_{\tau-1}), t_i) \right) = E_{x,t}(\nabla G(F(x, w_{\tau-1}), t)) = \nabla E_{x,t}(G(F(x, w_{\tau-1}), t))$$

Dans le cas d'apprentissage en-ligne on utilise souvent l'algorithme de descente de gradient stochastique qui est une adaptation de (2.1) : au lieu de sommer sur l'ensemble des formes, on tire à chaque étape un exemple  $x_\tau$  ce qui donne :

$$w_\tau = w_{\tau-1} - \eta_\tau \nabla G(F(x_\tau, w_{\tau-1}), t_\tau) \quad (2.6)$$

L'avantage majeur de cet algorithme est qu'il ne demande ni de connaître la densité de probabilité  $p(x, t)$  ni de l'estimer sur un échantillon.

Pour fixer le paramètre  $\eta_\tau$  il existe généralement deux procédés :

- Le cas à pas fixe. Le pas de gradient possède une norme faible, mais reste constant. C'est le cas pour les algorithmes adaptatifs.
- Le cas à pas décroissant. La norme du pas de gradient  $\eta_\tau$  décroît au cours de l'apprentissage. On dispose dans ce cas de théorèmes généraux de convergence.

### 2.2.3.3 Problèmes de régularité

$G$  est souvent une fonction continue dérivable mettant en relation  $F$  et  $t$ . Pour pouvoir assurer la convergence de l'algorithme de descente de gradient, on distingue deux cas :

- Le cas où  $F$  possède une différentielle intégrable, les propriétés générales des intégrales de Lebesgue assure que :

$$\forall w \quad E_{x,t}(\nabla G(F(x, w), t)) = \nabla E_{x,t}(G(F(x, w), t)) = \nabla C$$

Ce qui présente une condition suffisante de convergence de l'algorithme de descente de gradient.

- Le cas où  $F$  n'est pas différentiable sur un ensemble de mesure nulle. Pour pouvoir appliquer le théorème de convergence, il suffit que les accroissements de  $F$  par rapport à  $w$  soient majorés au voisinage de tout point  $(x, w)$  par une fonction intégrable  $\Gamma(x, w)$ .

$$\forall x, w \quad \forall h \in V(0) \quad \frac{1}{|h|} (F(x, w+h) - F(x, w)) < \Gamma(x, w)$$

Cette condition permet de conclure dans la plupart des cas.



### 2.2.4 Etude de la convergence

L'étude de la convergence peut se faire au moins par trois méthodes différentes :

1. Dans l'étude ordinaire, on suppose que l'on dispose d'un échantillon fini d'exemples étiquetés  $D_l$ . On applique l'algorithme stochastique (2.6) à des exemples tirés au hasard dans cet ensemble. On étudie ainsi la convergence de l'algorithme, appliquée à la minimisation d'une fonction empirique  $\bar{C}$  définie sur  $D_l$  avec une probabilité  $\bar{p}(x, t)$  discrète.

$$\bar{C} = \iint G(F(x, w), t) \bar{p}(x, t) dx dt = \frac{1}{n} \sum_{i=1}^n G(F(x_i, w), t_i)$$

On désire ici prouver que notre algorithme arrive à apprendre sur les  $n$  exemples dont nous disposons.

2. Dans l'étude asymptotique, on suppose que l'on tire un exemple issu d'une distribution de probabilité inconnue  $p(x, t)$ . On désire alors montrer que l'algorithme stochastique (2.6) converge vers un minimum de la fonction réelle  $C$ . L'étude asymptotique permet d'appliquer des raisonnements statistiques pour déterminer les propriétés des systèmes d'apprentissage [LeC86]. Mais les hypothèses asymptotiques sont loin de la réalité : on ne dispose ni d'un temps infini ni d'un nombre d'exemples suffisamment grand.

3. Dans l'étude de la généralisation, on suppose que l'on tire par avance un ensemble fini d'exemples, et que l'on applique l'algorithme stochastique (2.6) à des exemples tirés au hasard dans cet ensemble. On se propose de tenir compte des limitations de ressources en temps et en nombre d'exemples en cherchant à montrer que la convergence de l'algorithme (2.6) vers un minimum de la fonction empirique  $\bar{C}$  s'accompagne de la convergence de  $C$  vers une valeur proche de son minimum. Dans le cas où les exemples étiquetés sont en nombre limité, ce qui ne permet pas d'évaluer le modèle d'une manière adéquate, on peut avoir recours à la «Cross-Validation» [STO78, WAH75] pour apprendre le modèle ; la base d'apprentissage est alors divisée en  $S$  sous-ensembles distincts en tirant au hasard les exemples pour chaque sous-ensemble.

Dans l'étude asymptotique comme dans l'étude de la généralisation, on étudie la convergence de l'algorithme vers un système ayant un comportement optimal pour l'ensemble des nouvelles situations pouvant apparaître selon la distribution  $p(x, t)$ .

### 2.2.5 Dilemme biais-variance

Le but de l'apprentissage est d'assurer la meilleure généralisation possible. En pratique, on cherche à adapter la complexité du système d'apprentissage à la difficulté du problème et à la quantité d'informations dont on dispose pour résoudre ce problème.

Assurer une bonne généralisation revient en général à trouver un compromis entre complexité et robustesse de l'algorithme [GEM92]. Plusieurs approches ont été développées pour s'attaquer à ce problème. Nous n'entrerons pas dans le détail de ce sujet, et nous nous contenterons de présenter le problème classique du dilemme biais-variance qui illustre partiellement la problématique de la généralisation.

Nous allons considérer, pour simplifier l'exposé, le cas d'un modèle de régression utilisant comme critère global le critère quadratique, c'est-à-dire la somme au carré de la distance euclidienne entre les sorties calculées et les désirées. Dans ce cas on peut écrire :

$$C = \frac{1}{2} \iint (F(x, w) - t)^2 \cdot p(x, t) \, dx \, dt = \frac{1}{2} \iint (F(x, w) - t)^2 \cdot p(t/x) \cdot p(x) \, dx \, dt \quad (2.7)$$

Posons:

$$E[t/x] = \int t \cdot p(t/x) \, dt$$

$$E[t^2/x] = \int t^2 \cdot p(t/x) \, dt$$

En ajoutant et soustrayant  $E[t/x]$  dans  $(F(x, w) - t)^2$  il vient :

$$\begin{aligned} (F(x, w) - t)^2 &= (F(x, w) - E[t/x] + E[t/x] - t)^2 \\ &= (F(x, w) - E[t/x])^2 + (E[t/x] - t)^2 + 2 \cdot (F(x, w) - E[t/x]) \cdot (E[t/x] - t) \end{aligned} \quad (2.8)$$

En substituant la formule (2.8) dans (2.7), le troisième terme de (2.8) disparaît comme une conséquence de l'intégration sur  $t$ . Le critère global peut alors s'écrire :

$$C = \frac{1}{2} \int (F(x, w) - E[t/x])^2 p(x) \, dx + \frac{1}{2} \int (E[t^2/x] - E[t/x]^2) p(x) \, dx \quad (2.9)$$

Le second terme de (2.9) est indépendant du modèle, ce terme représente le bruit intrinsèque aux données. Ainsi le modèle optimal au sens de ce critère d'erreur est celui qui satisfait la relation  $F(x, w) = E[t/x]$ .

En pratique nous traitons avec des problèmes issus d'un ensemble de données de taille finie. Considérons tous les ensembles de données possibles contenant chacun  $N$  données, et issus de la même distribution de probabilité jointe  $p(x, t)$ . Nous avons déjà vu que le modèle optimal est celui qui prédit la moyenne conditionnelle  $E[t/x]$ . Une mesure qui permet de constater de combien la sortie  $F(x, w)$  du modèle est proche de cette moyenne conditionnelle est donnée par l'intégrale du premier terme de (2.9) :

$$(F(x, w) - E[t/x])^2 \quad (2.10)$$

La valeur de cette quantité peut être estimée sur une base d'apprentissage particulière  $D$  sur laquelle le modèle est entraîné, mais elle est alors très dépendante de l'ensemble  $D$  choisi. Nous pouvons éliminer cette dépendance en considérant la moyenne sur l'ensemble des bases d'apprentissage :

$$E_D[(F(x, w) - E[t/x])^2] \quad (2.11)$$

Où  $E_D[\cdot]$  indique l'espérance sur  $D$ . Notons que cette expression dépend de  $x$ . Si le modèle est un parfait estimateur de la fonction de régression  $E[t/x]$ , cette erreur est nulle. Deux raisons peuvent expliquer que ce ne soit pas le cas :

- Il se peut que la sortie du modèle soit en moyenne différente de la fonction de régression. Ceci est appelé le *biais*.
- Il se peut aussi que le modèle soit très sensible à un ensemble particulier de données  $D$ . Ainsi pour un  $x$  donné, la sortie de ce modèle peut être plus grande que la valeur attendue pour un certain ensemble de données ou plus petite pour un autre ensemble. Ceci est appelé la *variance*.

Nous pouvons faire apparaître analytiquement ces termes de biais et de variance dans l'expression de (2.11). Pour cela nous allons d'abord soustraire et ajouter  $E_D[F(x,w)]$  dans (2.10) :

$$\begin{aligned} (F(x,w) - E[t/x])^2 &= (F(x,w) - E_D[F(x,w)] + E_D[F(x,w)] - E[t/x])^2 \\ &= (F(x,w) - E_D[F(x,w)])^2 + (E_D[F(x,w)] - E[t/x])^2 \\ &\quad + 2(F(x,w) - E_D[F(x,w)]) \cdot (E_D[F(x,w)] - E[t/x]) \end{aligned}$$

Comme l'espérance  $E_D[F(x,w) - E_D[F(x,w)]]$  est nulle et que  $(E_D[F(x,w)] - E[t/x])$  est indépendant des données de  $D$ , nous pouvons écrire finalement :

$$E_D[(F(x,w) - E[t/x])^2] = \underbrace{(E_D[F(x,w)] - E[t/x])^2}_{\text{(biais)}^2} + \underbrace{E_D[(F(x,w) - E_D[F(x,w)])^2]}_{\text{variance}} \quad (2.12)$$

En regardant de plus près, le biais mesure de combien la sortie du modèle moyen diffère de la fonction désirée  $E[t/x]$ . Et la variance mesure la sensibilité de cette sortie  $F(x,w)$  à l'ensemble des données d'apprentissage.

Notons que les expressions du biais et de la variance sont des fonctions de l'entrée  $x$ . La signification du biais et de la variance apparaît plus clairement dans le problème de régression suivant. Supposons que les données soient générées par une fonction continue dérivable  $h(x)$  plus un bruit aléatoire additionnel  $\varepsilon$ , nul en moyenne, i.e.  $E_D[\varepsilon]=0$ .

$$t = h(x) + \varepsilon.$$

La fonction de régression dans ce cas est  $E[t/x] = h(x)$ . Un premier choix de modèle pour  $F(x,w)$  peut être une fonction  $f(x)$  complètement indépendante de l'ensemble des données  $D$  (figure 14). Dans ce cas, comme  $E_D[F(x,w)] = f(x) = F(x,w)$ , le terme de variance dans (2.12) disparaît. Toutefois, le terme de biais est très élevé car on n'a attaché aucune importance aux données. Le choix opposé consiste à utiliser une fonction qui s'ajuste parfaitement aux données (figure 15). Dans ce cas, le terme de biais dans (2.12) disparaît car :

$$E_D[F(x,w)] = E_D[h(x) + \varepsilon] = h(x) = E[t/x]$$

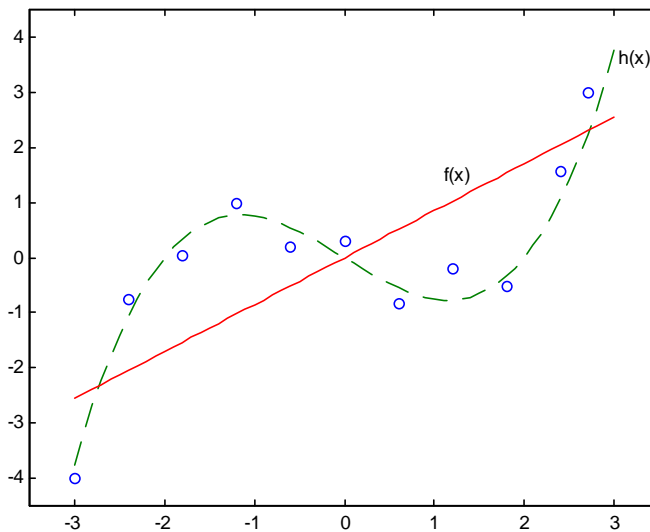


Figure 14. Une illustration schématique de la notion de biais et de variance. Les cercles représentent un ensemble de points générés par une fonction  $h(x)$  (en pointillée) avec du bruit additionnel. Le but est d'approximer  $h(x)$  le plus près que possible. Si l'on modélise les données par une fonction fixe  $f(x)$ , le biais est très élevé tandis que la variance est nulle.

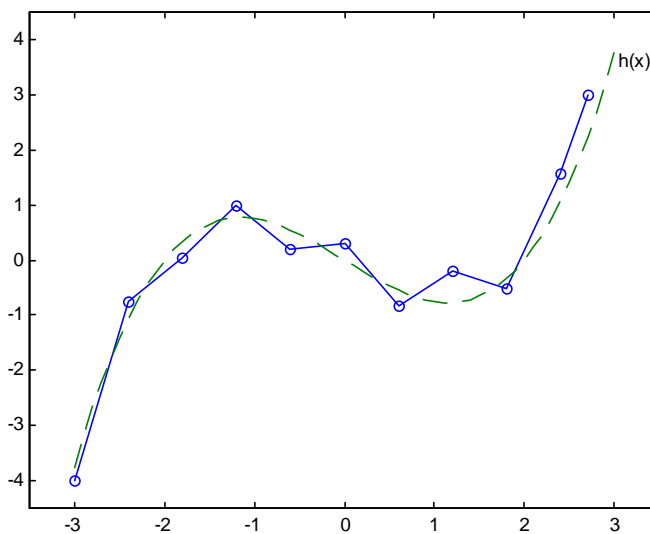


Figure 15. Comme dans la figure 14, mais le modèle utilisé est un interpolant exact des données. Dans ce cas le biais est faible mais la variance est élevée.

La variance est en contrepartie très élevée du fait que :

$$E_D[(F(x,w) - E_D[F(x,w)])^2] = E_D[F(x,w) - h(x)] = E_D[\varepsilon^2]$$

Ce terme représente en fait la variance du bruit sur les données et il peut être très élevé.

Nous constatons qu'il existe un compromis entre le biais et la variance. Une fonction qui s'ajuste aux données tend à avoir une grande variance et donne ainsi une grande erreur. Nous pouvons faire décroître la variance en lissant la fonction de prédiction mais si nous allons trop loin dans ce procédé le biais devient trop grand et l'erreur sera encore élevée.

Contrairement au cas de la régression, où la décomposition de l'erreur en deux termes de biais et de variance est unanimement utilisée, dans le cas de la discrimination il existe différentes versions de cette décomposition [KON95, KOH96, BRE96a] parmi lesquelles celle proposée dans [KON95] est la plus populaire.

### 2.2.6 Le vote majoritaire

Une des voies explorées depuis plusieurs années pour améliorer les capacités de généralisation des systèmes d'apprentissage consiste à combiner différents systèmes pour une même tâche. Nous allons nous intéresser ici à une famille de techniques qui font coopérer des systèmes identiques entraînés sur des ensembles d'apprentissage différents. Elles ont été développées récemment, vers le milieu des années 90, et ont connu un succès notable en apprentissage. Les deux représentants les plus connus sont le *Bagging* et le *Boosting*.

Notre intérêt pour ces techniques naît du fait que le Boosting a été employé avec succès en classification de textes [SCH00] et que des algorithmes dérivés comme le Co-Boosting présentent des similarités avec les algorithmes semi-supervisés que nous étudierons dans le chapitre 3.

Plusieurs appellations sont utilisées pour désigner les algorithmes de Boosting et de Bagging. Dans la littérature on parle soit d'algorithmes à base de vote (*Voting algorithms*) soit de moyennes pondérées (*weighted average algorithms*). Nous nous conformerons à la terminologie utilisée par les auteurs. Ces méthodes comparées à celles sans vote réduisent d'une manière significative l'erreur en généralisation. Dans la suite, nous allons étudier présenter ces deux types d'algorithmes et étudier leurs comportements en nous appuyant sur l'importance des termes de biais et de variance.

Les méthodes de vote peuvent être divisées en deux grandes familles : celles qui changent la distribution de probabilité  $p(x)$  de l'ensemble d'apprentissage en se basant sur le comportement des classifieurs précédents (les méthodes de *Boosting*) et celles qui ne le font pas (comme le *Bagging*). Chaque méthode utilise un algorithme de base (*weak algorithm*) et une base d'apprentissage. Le principe commun consiste à faire tourner cet algorithme plusieurs fois. Dans le cas du Boosting, la distribution des exemples de la base d'apprentissage est changée à chaque itération, tandis que dans le cas du Bagging les différentes bases d'apprentissage sont formées à l'avance. Les classifieurs générés sur chaque base sont alors combinés pour créer un classifieur final qui est utilisé sur la base de test.

#### 2.2.6.1 Bagging

L'algorithme de Bagging (**B**ootstrap **a**ggregating) a été introduit par Breiman [BRE96b]. Cet algorithme se déroule en deux étapes :

1. Par bootstrap, on génère d'abord  $B$  bases,  $D_1^1, D_1^2, \dots, D_1^B$  à partir de la base d'apprentissage initiale  $D_1$ , (un ensemble de bootstrap [EFR93] est généré en échantillonnant  $N$  exemples d'une base d'apprentissage par un tirage avec remise).
2. On entraîne ensuite  $B$  classifieurs  $Cl_1, Cl_2, \dots, Cl_B$  sur chacune de ces bases. Le classifieur final  $Cl^*$  est construit en combinant les  $Cl_i$ , la classe prédite par la combinaison est celle qui est majoritaire parmi ces classifieurs de base.

L'algorithme 5 illustre ce principe. Pour une base de Bootstrap donnée, un exemple issu de la base d'apprentissage originale (de taille  $N$ ) a une probabilité de  $1-(1-1/N)^N$  d'être sélectionné aléatoirement au moins une fois. Pour  $N$  assez grand, cette probabilité est de l'ordre de  $1-1/e = 63.2\%$ . Ce qui signifie que chaque base de bootstrap contient seulement 63.2% d'exemples de la base d'apprentissage apparaissant au moins une fois. Dans le cas d'algorithmes de base instables comme la descente de gradient stochastique, si les classifieurs ne sont pas corrélés la performance globale peut être améliorée [BRE94]. Dans le cas d'algorithmes stables la performance globale peut se dégrader dans le cas où la taille de la base d'apprentissage serait trop limitée [BRE96b].

---

**Entrée :** Une base d'apprentissage  $D_1$ , un modèle de base  $M_w$ , un entier  $B$  (nombre d'échantillons bootstrapés).

Pour  $i = 1$  à  $B$  faire

- {
1.  $D_1^i =$  échantillon bootstrapé de  $D_1$  (tirage avec remise)
  2.  $Cl_i = M_w(D_1^i)$
- }

$$Cl^*(x) = \arg \max_{t \in T} \sum_{i: Cl_i(x)=t} 1 \quad (\text{l'étiquette la plus souvent prédite})$$

**Sortie :** classifieur  $Cl^*$

---

#### Algorithme 5. L'algorithme de Bagging.

Breiman [BRE96b] explique que l'amélioration des performances est due en partie à la diminution de la variance. Mais cette affirmation a fait l'objet de controverses notamment par [SCH97]. Il existe néanmoins un excellent papier de [BAU99] qui tire la même conclusion que Breiman. Il s'agit donc encore aujourd'hui d'une question ouverte.

#### 2.2.6.2 Boosting

Schapiro a introduit l'algorithme de Boosting au début des années quatre-vingt dix [SCH90], cet algorithme est considéré comme l'une des méthodes les plus performantes pour accroître les performances d'un classifieur de base. Après les améliorations

apportées par Freund [FRE90], Freund et Schapire [FRE95] ont proposé l'algorithme d'AdaBoost (**Adaptive Boosting**) qui est l'un des algorithmes de boosting les plus populaires aujourd'hui. Dans cette partie nous allons nous intéresser aux deux variantes de cet algorithme : AdaBoost.M1 et AdaBoost.M2 ([FRE96]).

Comme le Bagging, l'algorithme d'AdaBoost génère un ensemble de classifieurs et les combine par une méthode à base de votes. A la différence du Bagging les classifieurs sont entraînés séquentiellement ; le  $i^{\text{ème}}$  classifieur généré prend en compte les erreurs des classifieurs déjà construits. Ceci peut être fait par exemple en ré-échantillonnant la base d'apprentissage : un exemple mal classé par un classifieur va être affecté d'un poids plus élevé qu'un exemple bien classé. De cette façon les exemples mal classés ont plus de chance d'être présents dans la nouvelle base d'apprentissage. Le but est de forcer le classifieur à se focaliser sur les exemples difficiles. Un premier algorithme adaboost est décrit ci-dessous (algorithme 6).

**Entrée** : Une base d'apprentissage  $D_l = \{x_i, t_i\}_{i=1, \dots, n}$ , un modèle de base  $M_w$ , un entier  $I$  (nombre d'itérations).

1.  $D_l^{(1)} = D_l$  avec le poids de chaque exemple égal à 1.

Pour  $i = 1$  à  $I$  faire

{

2.  $Cl_i = M_w(D_l^{(i)})$  (Apprendre les paramètres du classifieur  $C_i$  à partir de la base  $D_l^{(i)}$ )

3.  $\epsilon_i = \frac{1}{n} \sum_{x_j \in D_l^{(i)} / Cl_i(x_j) \neq t_j} \text{poids}(x_j)$  (Erreur pondérée sur la base  $D_l^{(i)}$ )

4. Si  $\epsilon_i \geq 1/2$ , poser  $i = I + 1$  et sortir de la boucle

5.  $\beta_i = \epsilon_i / (1 - \epsilon_i)$

6.  $\forall x_j \in D_l^{(i)}$ , si  $Cl_i(x_j) \neq t_j$  alors  $\text{poids}(x_j) = \text{poids}(x_j) \cdot \beta_i$

7. Construire la distribution de probabilité  $p(x)$  à partir du poids des exemples.

8. Construire la base  $D_l^{(i+1)}$  en tirant les exemples suivant la distribution  $p(x)$ .

}

$$Cl^*(x) = \arg \max_{t \in T} \sum_{i: C_i(x)=t} \log \frac{1}{\beta_i}$$

**Sortie** : classifieur  $Cl^*$

#### Algorithme 6. L'algorithme d'AdaBoost (M1)

Le principe de cet algorithme est le suivant : pour un nombre d'itérations  $I$ , on construit  $I$  bases d'apprentissage pondérées  $D_l^1, \dots, D_l^I$  et  $I$  classifieurs  $Cl_1, Cl_2, \dots, Cl_I$ . Le classifieur final  $Cl^*$  est formé en utilisant un schéma de vote pondéré : le poids de chaque classifieur dépend de ses performances sur la base d'apprentissage utilisée pour l'entraîner. Les deux propriétés fondamentales de l'algorithme AdaBoost.M1 sont :

1. Les exemples mal classés à l'étape  $i$  sont pondérés par un facteur inversement proportionnel à l'erreur du classifieur sur la base d'apprentissage, i.e.,  $1/(2\varepsilon_i)$ .
2. La proportion des exemples mal classés est  $\varepsilon_i$ , et le poids de ces exemples est amplifié par un facteur  $1/(2\varepsilon_i)$ . Ainsi le poids total des exemples mal classés après mise à jour sera la moitié du poids des exemples de la base d'apprentissage original. On peut ainsi avoir peu d'exemples mal classés ayant des poids élevés.

Le problème avec cet algorithme provient du fait que dès que le classifieur de base réalise une erreur  $\varepsilon_i \geq 1/2$  avec une base d'apprentissage donnée, l'algorithme s'arrête. Or ce cas de figure se manifeste fréquemment avec un mauvais classifieur ou avec une base d'apprentissage bruitée.

**Entrée :** Une base d'apprentissage  $D_I = \{x_i, t_i\}_{i=1, \dots, n}$ , un modèle de base  $M_w$ , un entier  $I$  (nombre d'itérations), posons  $\Psi = \{(k, t) / k \in \{1, \dots, n\}, t \neq t_k\}$ , l'ensemble des exemples mal classés.

Initialiser la distribution  $Dis_1(k, t) = 1/|\Psi|$  pour tout  $(k, t) \in \Psi$

Pour  $i = 1$  à  $I$  faire

{

1. Appeler le modèle  $M_w$  en le munissant de la distribution  $Dis_i$ .
2. Prendre l'hypothèse  $h_i : X \times T \rightarrow [0, 1]$ . (estimation de la probabilité a posteriori)
3. Calculer la pseudo-perte de  $h_i$  :

$$\varepsilon_i = \frac{1}{2} \sum_{(k, t) \in \Psi} Dis_i(k, t) (1 - h_i(x_k, t_k) + h_i(x_k, t))$$

4.  $\beta_i = \varepsilon_i / (1 - \varepsilon_i)$
5. Mettre à jour  $Dis_i$  :

$$Dis_{i+1}(k, t) = \frac{Dis_i(k, t)}{Z_i} \cdot \beta_i^{(1/2)(1+h_i(x_k, t_k)-h_i(x_k, t))}$$

Où  $Z_i$  est la constante de normalisation (de façon à ce que  $Dis_{i+1}$  soit une distribution)

}

$$h_{fin}(x) = \arg \max_{t \in T} \sum_{i=1}^I \left( \log \frac{1}{\beta_i} \right) h_i(x, t)$$

**Sortie :** classifieur  $h_{fin}$

#### Algorithme 7. L'algorithme d'AdaBoost (M2)

Pour résoudre ce problème Freund et Schapire [FRE96] ont proposé un deuxième algorithme, AdaBoostM.2, (voir algorithme 7).



A chaque itération  $i$  l'algorithme d'AdaBoost.M2 génère un classifieur  $h : X \times T \rightarrow [0,1]$  qui donne une estimation de la probabilité a posteriori des classes  $t$  sachant la forme d'entrée  $x$ . Les sorties du classifieur  $h$  sont interprétées de la manière suivante :

- Si  $h_i(x_k, t_k) = 1$  et  $h_i(x_k, t) = 0, \forall t \neq t_k$ , alors  $h_i$  a prédit correctement l'étiquette de  $x_k$ .
- Si  $h_i(x_k, t_k) = 0$  et  $h_i(x_k, t) = 1$ , alors  $h_i$  a donné la prédiction opposée.
- Si  $h_i(x_k, t_k) = h_i(x_k, t)$  alors la classe de  $x_k$  est choisie d'une manière aléatoire entre  $t$  et  $t_k$ .

Cette interprétation a conduit à la définition de la *pseudo-perte* (*pseudo-loss*) du classifieur  $h_i$  en regard de la distribution  $Dis_i$  définie par :

$$\varepsilon_i = \frac{1}{2} \sum_{(k,t) \in \Psi} Dis_i(k,t)(1 - h_i(x_k, t_k) + h_i(x_k, t)) \quad (2.13)$$

où  $\Psi$  est l'ensemble des exemples mal classés. Ainsi, au lieu d'utiliser la prédiction de l'erreur usuelle, la distribution de probabilité  $p(x)$  est obtenue sur la pseudo-perte. En utilisant ce critère, l'algorithme de Boosting se focalise non seulement sur les exemples difficiles mais plus spécifiquement sur les classes qui sont plus difficiles à discriminer.

Chaque paire mal classée  $(k,t)$  est intuitivement interprétée comme la réponse à une question binaire de la forme : « Est-ce que l'étiquette associée à l'exemple  $x_k$  est  $t_k$  (l'étiquette correcte) ou  $t$  (l'étiquette incorrecte) ? » Avec cette interprétation, le poids  $Dis_i(k,t)$  donne l'importance d'une étiquette incorrecte  $t$  pour l'exemple  $x_k$ .

Un résultat surprenant des algorithmes de Boosting est que l'erreur en généralisation décroît même lorsque l'erreur en apprentissage devient nulle. En utilisant la décomposition de l'erreur en biais et en variance, [SCH97] montrent que l'algorithme de Boosting combine deux effets :

1. il réduit le biais du classifieur de base en le forçant à se concentrer sur différentes parties de l'espace des exemples,
2. il réduit aussi la variance du classifieur de base en moyennant plusieurs hypothèses générées sur différents sous-ensemble de la base d'apprentissage.

Mais ceci n'explique pas la décroissance de l'erreur en généralisation après l'annulation de celle en apprentissage. On peut y voir plus clair en étudiant le comportement des algorithmes en terme de marges. La marge d'un exemple  $x_k$  est définie suivant :

$$\text{marge}(h, x_k, t_k) = h(x_k, t_k) - \max_{t \neq t_k} h(x_k, t) \quad (2.14)$$

Schapire et al. [SCH97] expliquent que la décroissance de l'erreur en généralisation est liée à la distribution de la marge des exemples en apprentissage. Ils démontrent qu'en se focalisant sur les exemples les plus difficiles, les algorithmes de Boosting tentent de maximiser la marge de tous les exemples en apprentissage.

En tenant compte soit de l'erreur soit de la pseudo-erreur les différences entre le Bagging et le Boosting peuvent se résumer comme suit :

1. Bagging génère des classifieurs de bases en parallèle tandis que Boosting les génère séquentiellement,

2. Le Bagging utilise le pré-échantillonnage et le Boosting la pondération,
3. Pour le classifieur final, le Bagging donne un poids égal à chaque classifieur de base, alors que le Boosting les pondère.

### 2.2.6.3 Un exemple : classification de décharges partielles

Nous avons étudié dans [AMI98] un système automatique de classification pour un problème de diagnostique de décharges partielles. Les décharges partielles se produisent lorsque la tension locale dans l'isolant dépasse une certaine tension seuil. Dans les systèmes électriques, elles comprennent une grande variété de phénomènes physiques, allant d'une simple émission de charges aux arcs électriques. Pendant des années, la reconnaissance de ces décharges se faisait en observant la trace de celles-ci sur un écran d'oscilloscope. Depuis peu, on a recours à des moyens informatiques plus avancés pour traiter ce problème.

Il s'agit d'un problème difficile car les moyens de détection sont très coûteux et il est difficile de faire des études complètes pour réunir des informations suffisantes pour caractériser ces décharges. Les prétraitements utilisés dans les différentes études sur ce sujet sont très variables (voir annexe) et ils n'ont jamais fait l'objet de comparaisons. Dans un premier temps nous avons cherché à évaluer la pertinence de ces différents prétraitements. Dans ce but nous avons conçu une méthode de validation basée sur le bootstrap [EFR93] qui nous a permis d'une part de comparer les différents prétraitements et d'autre part de les combiner. Une des conclusions de cette étude est que les meilleurs résultats sont obtenus lorsque tous les prétraitements sont combinés [AMI98]. C'est pour cela que dans les études que nous décrivons dans la suite de ce paragraphe, nous avons utilisé tous les prétraitements simultanément.

Nous nous sommes intéressé à comparer le Bagging et le Boosting pour ce problème de classification de décharges partielles. La base d'apprentissage que nous avons utilisée est constituée de 3 classes de décharges : les effets de couronnes (23% de la base), les effets de surface (24%), les arcs électriques (30.4%) ainsi que le bruit du site (22.6%). Nous avons ici utilisé l'ensemble des caractéristiques, une décharge est alors décrite par un vecteur de dimension 46. Avec un PMC de base nous avons obtenu une erreur de classification en test de 5.25%. La figure 16 haut, montre la combinaison après vote de ce classifieur en utilisant les algorithmes de Bagging et de Boosting.

Après 100 combinaisons, le Bagging ramène cette erreur à 3.425% tandis que le Boosting fait descendre cette erreur en dessous de 1.8%. Par ailleurs l'erreur en apprentissage pour l'algorithme de Boosting atteint 0 après seulement 8 itérations.

Nous avons étudié la distribution cumulée de la marge sur l'ensemble d'apprentissage. Pour visualiser cette distribution, [SCH97] mettent en relation la fraction d'exemples dont la marge est au plus  $x$  comme une fonction de  $x \in [-1, 1]$ . Sur la figure 16 bas, nous avons montré les graphes de distribution des marges correspondants aux expériences décrites ci-dessus. Nous observons que le Boosting et le Bagging tendent à augmenter les marges associées aux exemples et convergent vers une distribution où la plupart des exemples ont de grandes marges. On voit que la convergence de la distribution de la marge après chaque itération est plus accentuée dans le cas de Boosting, ceci s'explique

par le fait que le Boosting se focalise sur les exemples mal classés, car ce sont ceux qui ont une marge négative.

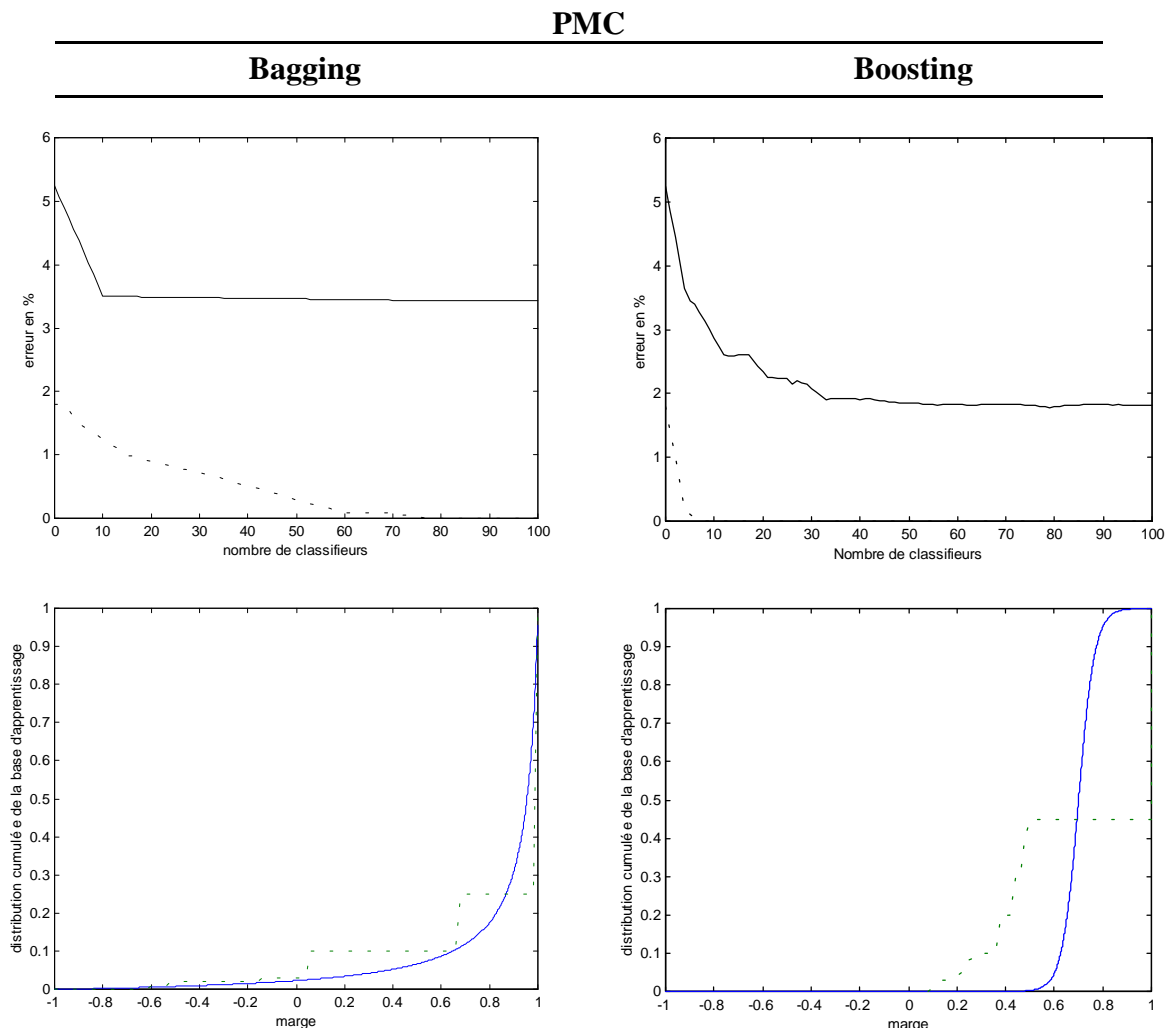


Figure 16. Courbes d'erreur (en apprentissage et en test) et distribution des marges pour le Boosting et le Bagging utilisant un PMC sur la base de décharges partielles. Les courbes du haut montrent les erreurs en apprentissage (trait pointillé) et en test (trait plein) en fonction du nombre de classifieurs combinés. Les courbes du bas montrent les distributions des marges cumulées de la base d'apprentissage, pour chacun de ces algorithmes après 2 (en trait pointillé) et 100 itérations (en trait plein).

## 2.3 Apprentissage Non-Supervisé

### 2.3.1 Introduction

Nous allons présenter dans cette section des procédures non-supervisées qui permettent d'apprendre à partir d'exemples non étiquetés. Il y existe trois raisons qui justifient ce genre d'apprentissage :

1. La collection et l'étiquetage d'un grand nombre d'exemples peuvent être coûteux en temps d'élaboration. Si un classifieur peut être initialisé sur un

petit ensemble d'exemples étiquetés, et qu'on l'utilise ensuite sans supervision sur un grand ensemble de formes non étiquetées, on peut gagner beaucoup de temps et d'investissement.

2. Dans beaucoup d'applications les caractéristiques des formes évoluent au cours du temps. L'apprentissage non-supervisé apporte alors une solution plus dynamique à ces problèmes que l'apprentissage supervisé.
3. Dans les premiers stades d'une étude il est parfois intéressant de connaître la nature ou la structure propre des données, (ceci en fonction des caractéristiques que nous optons pour décrire les exemples et de la distance que nous prenons pour les relier). La découverte des sous-classes distinctes peut aider à choisir le modèle adéquat.

La réponse à la question : « *Est-il possible d'apprendre quelque chose d'un exemple non étiqueté ?* » dépend des hypothèses de travail. L'apprentissage non-supervisé est souvent traité comme un problème d'estimation de densité. Malheureusement, c'est un problème difficile en particulier en grande dimension comme dans le cas des données textuelles. Cela a amené les chercheurs à reformuler le problème sous la forme d'un problème de partitionnement (ou *clustering*) des données en sous-groupes (ou *clusters*), qui est plus simple.

Nous n'allons pas donner dans cette section une description exhaustive des techniques non-supervisées. Nous nous focaliserons sur deux algorithmes de ce type, utilisés pour la fouille de données textuelles. Nous commençons par formuler le problème en termes de mélange de densités et de fonction de vraisemblance.

### 2.3.2 Mélange de densités

Un *mélange de densités* est une distribution de la forme :

$$p(x/\theta) = \sum_{k=1}^c \pi_k \cdot p(x/P_k, \theta_k) \quad (2.15)$$

où  $c$  est le nombre de composantes du mélange,  $\pi_j$  les différentes proportions (ou probabilités a priori) et  $p(x/P_k, \theta_k)$  les densités conditionnelles. Il y a trois ensembles de paramètres à estimer : les valeurs des  $\pi_k$ , le nombre de composantes et les paramètres du mélange. Etant données  $m$  observations  $X = (x_1, \dots, x_m)$ , le logarithme de la fonction de vraisemblance du mélange est :

$$L_M(\Theta) = \log p(X/\Theta) = \sum_{i=1}^m \log \left( \sum_{k=1}^c \pi_k \cdot p(x_i/P_k, \theta_k) \right) \quad (2.16)$$

où  $\Theta$  dénote l'ensemble des paramètres du modèle  $\{\pi_1, \dots, \pi_c; \theta_1, \dots, \theta_c\}$ . Dans le cas le plus courant :

1. Les exemples proviennent d'un nombre connu  $c$  de classes,
2. Les probabilités a priori  $\{\pi_k\}_{k=1, \dots, c}$  de chacun des classes sont connues,

3. Les formes des densités conditionnelles  $\{p(x/P_k, \theta_k)\}_{k=1,\dots,c}$  sont connues,
4. Les inconnues sont les  $c$  valeurs des vecteurs de paramètres  $\theta = \{\theta_k\}_{k=1,\dots,c}$

De plus on suppose que chaque exemple  $x$  appartient à une seule partition  $P_k$  (de probabilité  $\pi_k$ ).

Le but est d'utiliser les exemples issus de ce mélange de densités pour estimer le vecteur de paramètres inconnu  $\theta$ . [DuH73] explicitent les cas dans lesquels il est possible de réaliser cette estimation. Ils introduisent la notion d'*identifiabilité* de  $p(x/\theta)$  ;  $p(x/\theta)$  est dit *identifiable* si pour tout  $\theta \neq \theta'$  il existe un  $x$  tel que  $p(x/\theta) \neq p(x/\theta')$ . Si  $p(x/\theta)$  est identifiable on peut donner une estimation de cette densité.

En général, la non-identifiabilité correspond aux distributions discrètes. Ainsi lorsqu'il y a trop de composantes dans le mélange, il peut y avoir plus d'inconnues que d'équations indépendantes et l'identifiabilité peut être un vrai problème. Pour le cas continu, les problèmes d'identifiabilités sont plus solvables. Dans la section 2.3.5, nous allons nous intéresser à l'estimation de paramètres dans le cas où à chaque exemple  $x$  est associée une croyance  $z$  ( $z \in [0,1]$ ) qui est l'hypothèse conjoncturée par le superviseur de la probabilité que  $x$  appartienne à une partition donnée. Auparavant, nous présentons les algorithmes EM et CEM.

### 2.3.3 Algorithme EM

#### 2.3.3.1 L'algorithme

Afin de déterminer les paramètres  $\Theta$  d'un modèle qui explique au mieux un ensemble d'observations  $X$ . On peut chercher à maximiser la vraisemblance du mélange (ou  $L_M$ ) suivant  $\Theta$ . En général, il n'est pas possible de résoudre explicitement :

$$\frac{\partial L_M}{\partial \Theta} = 0 \quad (2.17)$$

Une manière de maximiser  $L_M$  est d'utiliser une classe générale de procédures itératives connues sous le nom d'algorithme EM [DEM77]. Cet algorithme est l'un des algorithmes d'optimisation les plus employés en statistiques. Parmi les applications classiques qui utilisent cet algorithme on peut citer les MMC et l'estimation des paramètres d'un mélange de densités. En général avec cet algorithme on ne peut trouver qu'un maximum local.

Le principe de cet algorithme est le suivant : A chaque itération, les valeurs de  $\Theta$  sont ré-estimées de façon à accroître  $L_M$  et ceci jusqu'à ce qu'un maximum soit atteint. [BIS95, McL97, MIT97, WEB99] ont étudié en détail les différents points de l'algorithme EM, nous n'allons nous intéresser ici qu'à sa forme générale et nous donnerons une preuve de sa convergence.

L'idée principale de l'algorithme EM est d'introduire des variables cachées  $Z$  de façon à ce que, si les  $Z$  étaient connus, la valeur optimale de  $\Theta$  pourrait être trouvée facilement. Ainsi, on peut écrire :

$$L_M(\Theta) = \log p(X / \Theta) = \log \sum_Z p(X, Z / \Theta) = \log \sum_Z p(X / Z, \Theta) p(Z / \Theta) \quad (2.18)$$

Où  $\sum_Z$  est la somme (ou intégrale) suivant les variables cachées  $Z$ .

En notant l'estimation courante des paramètres à l'itération  $j$ ,  $\Theta^{(j)}$ , l'itération  $j+1$  consiste à trouver une nouvelle estimation des paramètres  $\Theta$  qui maximise :

$$L_M(\Theta) - L_M(\Theta^{(j)}) = \log p(X/\Theta) - \log p(X/\Theta^{(j)}) = \log \frac{p(X/\Theta)}{p(X/\Theta^{(j)})} \quad (2.19)$$

Soit d'après (2.18) :

$$L_M(\Theta) - L_M(\Theta^{(j)}) = \log \frac{\sum_Z p(X/Z, \Theta) p(Z/\Theta)}{p(X/\Theta^{(j)})} \quad (2.20)$$

On peut réécrire l'équation (2.20) comme :

$$L_M(\Theta) - L_M(\Theta^{(j)}) = \log \sum_Z p(Z/X, \Theta^{(j)}) \cdot \frac{p(X/Z, \Theta) p(Z/\Theta)}{p(Z/X, \Theta^{(j)}) p(X/\Theta^{(j)})}$$

En utilisant la concavité de la fonction logarithme on peut majorer le logarithme d'une somme par la somme des logarithmes (inégalité de Jensen):

$$\log \sum_i \lambda_i y_i \geq \sum_i \lambda_i \log y_i, \text{ si } \sum_i \lambda_i = 1$$

Ainsi en prenant  $\lambda_Z = p(Z/X, \theta_i)$  il vient :

$$L_M(\Theta) - L_M(\Theta^{(j)}) \geq \sum_Z p(Z/X, \Theta^{(j)}) \cdot \log \frac{p(X/Z, \Theta) p(Z/\Theta)}{p(X/\Theta^{(j)}) p(Z/X, \Theta^{(j)})}$$

Ce qui peut être réécrit comme  $L_M(\Theta) \geq Q(\Theta, \Theta^{(j)})$ , où

$$Q(\Theta, \Theta^{(j)}) = L_M(\Theta^{(j)}) + \sum_Z p(Z/X, \Theta^{(j)}) \cdot \log \frac{p(X/Z, \Theta) p(Z/\Theta)}{p(X/\Theta^{(j)}) p(Z/X, \Theta^{(j)})}$$

Ainsi  $L_M(\Theta)$  est égal à  $Q(\Theta, \Theta^{(j)})$  pour  $\Theta = \Theta^{(j)}$  et il est strictement plus grand que  $Q(\Theta, \Theta^{(j)})$  partout ailleurs. La figure 17 illustre ce fait. A l'étape  $j+1$ , nous cherchons une nouvelle valeur de  $\Theta$  qui maximise  $Q(\Theta, \Theta^{(j)})$  soit :

$$\begin{aligned} \Theta^{(j+1)} &= \arg \max_{\Theta} \left[ L_M(\Theta^{(j)}) + \sum_Z p(Z/X, \Theta^{(j)}) \cdot \log \frac{p(X/Z, \Theta) p(Z/\Theta)}{p(X/\Theta^{(j)}) p(Z/X, \Theta^{(j)})} \right] \\ &= \arg \max_{\Theta} \left[ \sum_Z p(Z/X, \Theta^{(j)}) \cdot \log [p(X/Z, \Theta) p(Z/\Theta)] \right] \\ &= \arg \max_{\Theta} E_{Z/X, \Theta^{(j)}} [p(X/Z, \Theta)] \end{aligned}$$

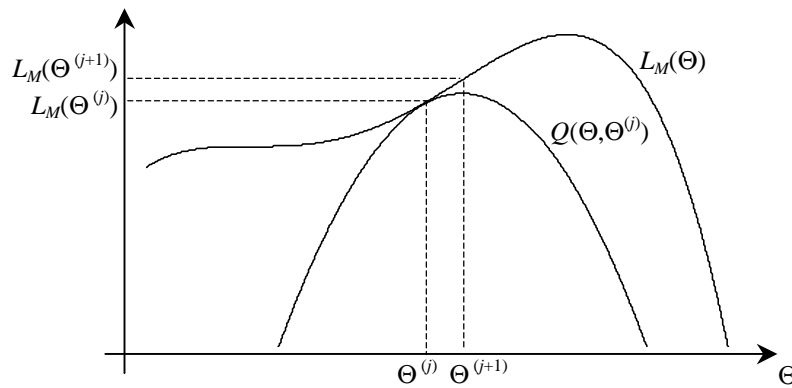


Figure 17. Une représentation graphique montrant le fonctionnement de l'algorithme EM. L'axe des abscisses représente les valeurs possibles de l'ensemble des paramètres  $\Theta$  et l'axe des ordonnées donne le logarithme de la vraisemblance des données. L'algorithme EM calcule la fonction  $Q(\Theta, \Theta^{(j)})$  en utilisant l'estimation courante  $\Theta^{(j)}$  et donne la nouvelle  $\Theta^{(j+1)}$  comme le point maximum de  $Q(\Theta, \Theta^{(j)})$ .

L'algorithme de EM peut être résumé par l'algorithme suivant :

---

Initialisation des paramètres  $\Theta^{(0)}$

Pour  $j \geq 0$ ,

Etape **E** : Calculer l'espérance  $E_{Z/X, \Theta^{(j)}} [p(X/Z, \Theta)]$ .

Etape **M** : Trouver  $\Theta^{(j+1)}$  qui maximise  $Q(\Theta, \Theta^{(j)})$ .

---

Algorithme 8. L'algorithme de EM

Dans la formulation précédente :

- $L_M(\Theta^{(j+1)}) \geq Q(\Theta^{(j+1)}, \Theta^{(j)}) \geq Q(\Theta^{(j)}, \Theta^{(j)})$ ,
- $L_M(\Theta^{(j)}) = Q(\Theta^{(j)}, \Theta^{(j)})$

donc à chaque itération on a

$$L_M(\Theta^{(j+1)}) \geq L_M(\Theta^{(j)})$$

La convergence réside donc simplement sur l'inégalité  $Q(\Theta^{(j+1)}, \Theta^{(j)}) \geq Q(\Theta^{(j)}, \Theta^{(j)})$ . Même si l'étape *M* ne peut pas se faire facilement et que le gradient de  $L_M$  ne peut pas se calculer simplement, il est encore possible de garantir la convergence en utilisant la version généralisée de EM appelée l'algorithme GEM [DEM77].

### 2.3.3.2 Un exemple : Estimation des paramètres d'un mélange gaussien

En annexe nous avons donné la preuve que, dans le cas où les données sont issues d'une distribution normale  $X \sim \mathcal{N}_p(\mu, \Sigma)$ , le logarithme de la vraisemblance basé sur les  $m$  observations  $X = (x_1, \dots, x_m)$ ,  $L_M(\mu, \Sigma)$ , est maximal lorsqu'on prend pour vecteur moyenne et matrice de covariance, la moyenne et la variance des  $m$  données [WAT64], i.e.

$$\mu = \bar{x} = \frac{1}{m} \sum_{i=1}^m x_i ,$$

$$\Sigma = Var = \frac{1}{m} \sum_{i=1}^m (x_i - \bar{x})(x_i - \bar{x})^t$$

Nous allons nous intéresser maintenant à un exemple d'application de l'algorithme EM pour estimer les paramètres d'un mélange de  $c$  densités gaussiennes  $\mathcal{N}_p(\mu_k, \Sigma_k)_{k=1, \dots, c}$  :

$$p(x / \theta) = \sum_{k=1}^c \pi_k \cdot p(x / P_k, \mu_k, \Sigma_k)$$

Trouver le maximum de la fonction de vraisemblance d'un mélange de  $c$  densités gaussiennes en regard des paramètres  $\mu_k$  et  $\Sigma_k$  revient à trouver les estimateurs du maximum de vraisemblance des paramètres de  $c$  densités normales indépendantes de taille  $m_1, \dots, m_c$  ( $m = m_1 + \dots + m_c$ ) [FLU97], soit :

$$\pi_k = \frac{1}{m} \sum_{i=1}^m t_{ki} = \frac{m_k}{m}$$

$$\forall k = \{1, \dots, c\}, \mu_k = \frac{1}{m_k} \sum_{i=1}^m t_{ki} \cdot x_i = \frac{1}{\pi_k \cdot m} \sum_{i=1}^m t_{ki} \cdot x_i,$$

$$\Sigma_k = \frac{1}{m_k} \sum_{i=1}^m t_{ki} (x_i - \mu_k)(x_i - \mu_k)^t = \frac{1}{\pi_k \cdot m} \sum_{i=1}^m t_{ki} (x_i - \mu_k)(x_i - \mu_k)^t$$

Où les  $t_{ki}$  sont les variables d'indicateur de classes :  $t_{ki} = 1 \Leftrightarrow x_i \in C_k$ . Le calcul des paramètres du mélange dépend de ces variables indicateurs. L'espérance des variables  $t_{ki}$  étant donné les exemples et les paramètres  $(\mu_k, \Sigma_k)_{k=1, \dots, c}$  est donnée par :

$$E[t_{ki} / X, \theta] = E[t_{ki} / x_i, \theta] = p(t_{ki} = 1 / x_i, \theta)$$

Soit d'après la règle de Bayes :

$$\pi_{ki} = E[t_{ki} / X, \theta] = p(t_{ki} = 1 / x_i, \theta) = \frac{\pi_k \cdot p(x_i / P_k, \theta)}{\sum_{k=1}^c \pi_k \cdot p(x_i / P_k, \theta)}$$

L'étape **E** consiste donc à remplacer les  $t_{ki}$  par leurs espérances  $\pi_{ki}$ . On peut donc donner l'algorithme EM pour l'estimation des paramètres de ce mélange, (algorithme 9).



---

Initialisation des paramètres  $\pi^{(0)}$ ,  $\mu^{(0)}$  et  $\Sigma^{(0)}$

Pour  $j \geq 0$ , jusqu'à la convergence faire :

Etape **E** : Calculer  $T^{(j)} = E[T / X ; \pi^{(j)}, \mu^{(j)}, \Sigma^{(j)}]$ , soit :

$$\pi_{ki}^{(j)} = E[t_{ki} / X, \theta^{(j)}] = \frac{\pi_k^{(j)} \cdot p(x_i / P_k^{(j)}, \theta^{(j)})}{\sum_{k=1}^c \pi_k^{(j)} \cdot p(x_i / P_k^{(j)}, \theta^{(j)})}, \forall i = 1, \dots, m ; \forall k = 1, \dots, c$$

Etape **M** : Trouver de nouvelles valeurs de paramètres  $\pi^{(j+1)}$ ,  $\mu^{(j+1)}$  et  $\Sigma^{(j+1)}$  qui maximisent le logarithme de la vraisemblance (soit d'après l'annexe) :

$$\pi_k^{(j+1)} = \frac{1}{m} \sum_{i=1}^m \pi_{ki}^{(j)}, \forall k = 1, \dots, c$$

$$\mu_k^{(j+1)} = \frac{1}{m \pi_k^{(j)}} \sum_{i=1}^m \pi_{ki}^{(j)} \cdot x_i, \forall k = 1, \dots, c$$

$$\Sigma_k^{(j+1)} = \frac{1}{m \pi_k^{(j)}} \sum_{i=1}^m \pi_{ki}^{(j)} \cdot (x_i - \mu_k^{(j)})(x_i - \mu_k^{(j)})^t, \forall k = 1, \dots, c$$

---

Algorithme 9. Algorithme EM pour l'estimation des paramètres d'un mélange de densités gaussiennes.

Dans cette approche, les paramètres  $\{\pi, \theta\}$  sont choisis de façon à maximiser le logarithme de vraisemblance (2.16). Si on recherche un partitionnement des données  $P=(P_1, \dots, P_c)$ , il peut alors être obtenu en déduisant directement à partir des estimateurs du maximum de vraisemblance, les probabilités a posteriori des partitions. On assigne ainsi à chaque exemple  $x_i$  la partition qui donne la plus grande probabilité a posteriori [FLU97, CEL93] :

$$E[t_{ki} / x_i, \theta, \pi] = p(t_{ki} = 1 / x_i, \theta, \pi) = \frac{\pi_k \cdot p(x_i / t_{ki}, \theta)}{\sum_{k=1}^c \pi_k p(x_i / t_{ki}, \theta)}$$

[NIG99] ont par exemple utilisé cette approche pour la classification de documents.

### 2.3.4 Maximum de vraisemblance de classification

Il existe une autre approche pour faire de la discrimination en mode non-supervisé appelé le *maximum de vraisemblance de classification*. Dans cette approche les vecteurs indicateurs de classes  $t_k$  sont traités comme des paramètres manquants et on maximise directement le logarithme de la vraisemblance de classification<sup>9</sup>.

---

<sup>9</sup> On donne en annexe les différentes étapes pour calculer cette vraisemblance

$$L_C(P, \pi, \theta) = \sum_{k=1}^c \sum_{i=1}^m t_{ki} \cdot \log(\pi_k \cdot p(x_i / P_k, \theta_k)) \quad (2.21)$$

Ce critère est maximisé grâce à l'algorithme de classification EM (CEM) [McL92, CEL92]. Il est similaire à l'algorithme EM excepté une étape **C** de classification dans laquelle, à chaque donnée, est assignée une et une seule composante du mélange, (algorithme 10).

---

**Initialisation:** Commencer avec une partition  $P^{(0)}$  initiale.

Pour  $j = 0$  jusqu'à convergence faire

- Etape **E** : Estimer avec les paramètres courants  $\{\pi^{(j)}, \theta^{(j)}\}$ , les probabilités a posteriori d'appartenance aux partitions  $P_k$  pour tout  $k=1, \dots, c$

$$E[t_{ki}^{(j)} / x_i; P^{(j)}, \pi^{(j)}, \theta^{(j)}] = \frac{\pi_k^{(j)} \cdot p(x_i / P_k^{(j)}, \theta_k^{(j)})}{\sum_{k=1}^c \pi_k^{(j)} \cdot p(x_i / P_k^{(j)}, \theta_k^{(j)})}$$

- Etape **C** : Assigner à chaque exemple  $x_i$  sa partition, celle dont la probabilité a posteriori est maximale. Noter  $P^{(j+1)}$  la nouvelle partition.
  - Etape **M** : Estimer les nouveaux paramètres  $\{\pi^{(j+1)}, \theta^{(j+1)}\}$  qui maximisent  $L_C(P^{(j+1)}, \pi^{(j)}, \theta^{(j)})$ .
- 

#### Algorithme 10. L'algorithme CEM

La convergence de cet algorithme est obtenue simplement grâce aux deux étapes **C** et **M**.

- A l'étape **C** on choisit, avec les paramètres courants  $\{\pi^{(j)}, \theta^{(j)}\}$ , une partition  $P^{(j+1)}$  suivant la règle optimale de Bayes, soit :

$$L_C(P^{(j+1)}, \pi^{(j)}, \theta^{(j)}) \geq L_C(P^{(j)}, \pi^{(j)}, \theta^{(j)})$$

- A l'étape **M** on cherche de nouveaux paramètres  $\{\pi^{(j+1)}, \theta^{(j+1)}\}$  qui maximisent  $L_C(P^{(j+1)}, \pi^{(j)}, \theta^{(j)})$ , soit :

$$L_C(P^{(j+1)}, \pi^{(j+1)}, \theta^{(j+1)}) \geq L_C(P^{(j+1)}, \pi^{(j)}, \theta^{(j)})$$

Et donc à chaque itération  $j$  on a :

$$L_C(P^{(j+1)}, \pi^{(j+1)}, \theta^{(j+1)}) \geq L_C(P^{(j)}, \pi^{(j)}, \theta^{(j)})$$

### 2.3.5 Estimation de paramètres dans le cas de supervision imparfaite

Nous allons nous intéresser dans cette section à une méthode d'estimation de paramètres dans le cas où l'on dispose d'une prédiction imparfaite de l'appartenance aux classes des exemples. Cette connaissance imparfaite peut provenir soit d'un étiquetage « à la main » peu précis, soit d'un premier système de classification

éventuellement peu performant. Développer des méthodes d'apprentissage utilisant ce type d'entrée permet de s'affranchir d'une phase d'étiquetage manuel précis d'une collection de données d'apprentissage. Ceci est particulièrement intéressant dans le cas de problèmes de fouille de données qui nécessitent généralement de très grosses bases de données qu'il est très coûteux et fastidieux d'étiqueter. Nous pensons donc que cette problématique, l'apprentissage à partir de données étiquetées imparfaitement, peut s'appliquer à une large gamme de problèmes de fouille de données textuelles.

Avec un modèle imparfait on peut notamment :

- donner un étiquetage partiellement incorrect des entités,
- améliorer cet étiquetage avec un modèle de mélange dont on estimera les paramètres par rapport aux données.

Dans ce cadre, Krishnan et Nardy [KRI87] et Titterington [TIT87] ont proposé une méthode d'estimation de paramètres d'un mélange de densités pour un problème de classification. Nous décrivons ici leurs travaux. Ils considèrent dans leur étude qu'à chaque forme est associée une étiquette incertaine, les vraies étiquettes des exemples étant inconnues. En particulier, [KRI87] considèrent en détail un problème à deux classes où à chaque forme  $x_i$  (de dimension  $p$ ) est associée l'hypothèse  $z_i$  ( $z_i \in [0,1]$ ).  $z_i$  représente la croyance du superviseur imparfait en l'appartenance de l'exemple  $x_i$  à la classe 1.

Les données sont supposées issues d'un mélange de densités (2.15) où les densités conditionnelles des classes sont de la forme :

$$p(x/z, \theta_k) = f_k(x).q_k(z), k=1,2,$$

où  $f_1(x)$ ,  $f_2(x)$  sont des densités normales  $\mathcal{N}_p(\mu_1, \Sigma)$ ,  $\mathcal{N}_p(\mu_2, \Sigma)$  et  $q_1(z)$ ,  $q_2(z)$  sont les densités Beta( $a,b$ ) et Beta( $b,a$ ). La multiplication des densités normales par des densités Beta permet d'estimer une large famille de distributions.

[KRI87] remarquent que ce problème est un problème avec données manquantes dans le sens où le vrai étiquetage  $t_k$  de l'exemple  $x_k$  est manquant. Ils appliquent alors l'algorithme EM pour trouver les estimations du maximum de vraisemblance de tous les paramètres  $\pi_1$ ,  $\pi_2$ ,  $\mu_1$ ,  $\mu_2$ ,  $\Sigma$ ,  $a$  et  $b$ . D'après leur analyse, tant que les paramètres  $m$  et  $n$  sont à évaluer, l'étape **M** de l'algorithme ne peut donner de solutions exactes.

[TIT87] donnent une solution à ce problème grâce à une analyse bayésienne et à une généralisation du modèle utilisé. Au lieu de prendre  $z$  comme l'hypothèse imparfaite du superviseur, [TIT87] suggèrent de prendre  $w = \log\{z/(1-z)\}$ . Dans ce cas l'espace des  $w$  est une droite réelle et la distribution de  $w$  pour chaque classe devient une distribution normale, cette forme est la version la plus simple de la distribution logistique-normale [AIT76]. [TIT87] supposent de plus que  $q_1(w)$  et  $q_2(w)$  sont symétriques de la forme  $\mathcal{N}_p(-\Delta, \Omega)$  et  $\mathcal{N}_p(\Delta, \Omega)$  où  $\Omega > 0$ . Dans ce cas, le mélange de densités en terme de  $x$  et  $w$  devient un mélange de deux densités normales avec une matrice de covariance commune. D'après la section 2.3.3, le maximum de vraisemblance pour un mélange de densités normales a des solutions exactes.

Nous allons présenter dans ce qui suit l'algorithme décrit dans [TIT87] utilisant l'algorithme EM pour l'estimation des paramètres de ce modèle, (algorithme 11). Cet algorithme présente des similarités avec l'algorithme 9. Nous posons  $\{(w_i, x_i)\}_{i=1, \dots, m}$  et  $\{(t_i, w_i, x_i)\}_{i=1, \dots, m}$  comme étant respectivement les données observées et les données complètes correspondantes.

---

Initialisation des paramètres  $\pi^{(0)}, \mu^{(0)}, \Sigma^{(0)}, \Delta^{(0)}, \Omega^{(0)}$

Pour  $j \geq 0$ , jusqu'à la convergence faire :

Etape **E** : Calculer:

$$\pi_{i1}^{(j)} = \frac{\pi_1^{(j)} f_1^{(j)}(x_i) q_1^{(j)}(w_i)}{\pi_1^{(j)} f_1^{(j)}(x_i) q_1^{(j)}(w_i) + (1 - \pi_1^{(j)}) f_2^{(j)}(x_i) q_2^{(j)}(w_i)}, \forall i = 1, \dots, m$$

où  $f_k^{(j)}, q_1^{(j)}$  et  $q_2^{(j)}$  sont les densités normales respectivement  $\mathcal{N}_p(\mu_k^{(j)}, \Sigma^{(j)})$ ,  $\mathcal{N}_p(-\Delta^{(j)}, \Omega^{(j)})$  et  $\mathcal{N}_p(-\Delta^{(j)}, \Omega^{(j)})$

Etape **M** : Trouver des valeurs nouvelles de paramètres  $\pi^{(j+1)}, \mu^{(j+1)}, \Sigma^{(j+1)}, \Delta^{(j+1)}, \Omega^{(j+1)}$  qui maximisent le logarithme de la vraisemblance:

$$\pi_1^{(j+1)} = m^{-1} \sum_{i=1}^m \pi_{i1}^{(j)}$$

$$\mu_1^{(j+1)} = (m \pi_1^{(j)})^{-1} \sum_{i=1}^m \pi_{i1}^{(j)} x_i$$

$$\mu_2^{(j+1)} = (m(1 - \pi_1^{(j)}))^{-1} \sum_{i=1}^m (1 - \pi_{i1}^{(j)}) x_i$$

$$\Sigma^{(j+1)} = m^{-1} \sum_{i=1}^m \left\{ \pi_{i1}^{(j)} (x_i - \mu_1^{(j)})(x_i - \mu_1^{(j)})^t + (1 - \pi_{i1}^{(j)}) (x_i - \mu_2^{(j)})(x_i - \mu_2^{(j)})^t \right\}$$

$$\Delta^{(j+1)} = m^{-1} \sum_{i=1}^m (2\pi_{i1}^{(j)} - 1) w_i$$

$$\Omega^{(j+1)} = m^{-1} \left[ \sum_{i=1}^m w_i^2 - m \Delta^{(j)2} \right]$$


---

Algorithme 11. Algorithme EM pour l'estimation de paramètres d'un mélange de densités dans le cas de supervision imparfaite.

## 2.4 Conclusion

Nous avons étudié dans ce chapitre quelques problématiques de l'apprentissage supervisé et non-supervisé. Comme les problèmes d'optimisation, de régularité et de généralisation dans le domaine de l'apprentissage supervisé. Et l'estimation de

paramètres du mélange de densités, les algorithmes EM et CEM ainsi que les critères de vraisemblance de mélange et de classification dans le cadre de l'apprentissage non-supervisé. Beaucoup de questions sont encore ouvertes aujourd'hui dans ces domaines et font l'objet de recherches. Par ailleurs, depuis quelques années, de nouvelles problématiques ne rentrant pas dans le cadre de ces deux types d'apprentissage commencent à émerger. Par exemple un problème de classification de pages *web* à deux classes *pertinent*, *non-pertinent* où on disposerait d'un petit ensemble de documents étiquetés et d'une large collection de documents non étiquetés. Cette problématique de l'apprentissage, connue sous le nom de *l'apprentissage semi-supervisé*, fait l'objet du chapitre suivant.

## 2.5 Bibliographie

- [AIT76] J. Aitchison, C.B. Begg. Statistical diagnosis when basic cases are not classified with certainty. *Biometrika*, vol. 93, pp.1--12, 1976.
- [AMI98] M.R. Amini, P. Gallinari, F. d'Alché-Buc, F. Bonnard, E. Fernandez. Automated statistical recognition of Partial Discharges in insulation systems. *In the Proceedings of the 8th International Conference of Artificial Neural Networks*, pp. 701--706, 1998.
- [BAU99] E. Bauer, R. Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants. *Machine Learning : Proceedings of the Sixteenth International Conference*, Kluwer Academic Publishers, N° 36, pp. 105--142, 1999.
- [BIS95] C. M. Bishop. Neural Networks for Pattern Recognition. *Clarendon Press*, Oxford, 1995.
- [BOT91] L. Bottou. Une approche théorique de l'apprentissage connexioniste; Application à la reconnaissance de la parole. *Thèse de doctorat*, Université de Paris XI, Orsay, 1991.
- [BRE94] L. Breiman. Heuristics of instability in model selection, *Technical Report*, université de Berkeley, 1994.
- [BRE96a] L. Breiman. Arcing classifiers, *Technical Report*, 1996.  
<http://www.stat.Berkley.EDU/users/breiman/>.
- [BRE96b] L. Breiman. Bagging predictors, *Machine Learning*, vol. 4, pp. 123--140, 1996.
- [CEL92] G. Celeux, G. Govaert. A Classification EM algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*. Vol. 14, pp. 351--332, 1992.
- [DEM77] A. Dempster, N. Laird, D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, Vol. B, N° 39, pp. 1--38, 1977.
- [DuH73] R. O. Duda, P. E. Hart. Pattern Classification and Scene Analysis. *John Wiley*, New York, 1973.
- [EFR93] B. Efron, R. Tibshirani. An Introduction to the Bootstrap, *Chapman & Hall*, 1993.
- [EVE84] B. S. Everitt. In Introduction to Latent Variable Models. *Chapman and Hall*, 1984.

- [FLU97] B. Flury, A First Course in Multivariate Statistics. *Springer-Verlag*, 1997.
- [FRE90] Y. Freund. Boosting a weak learning algorithm by majority. *Proceedings of the third Annual Workshop on Computational Learning Theory*, pp. 202--216, 1990.
- [FRE95] Y. Freund, R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Proceedings of the second European Conference on Computational Learning Theory*. Springer-Verlag, pp. 23-37, 1995.
- [FRE96] Y. Freund, R. E. Schapire. Experiments with a new boosting algorithm. *Machine Learning : Proceedings of the Thirteenth International Conference*, Morgan Kauffmann, pp. 148--156, 1996.
- [GEM92] S. Geman, E. Bienenstock, R. Doursat. Neural Networks and the bias / variance dilemma. *Neural Computation*, Vol. 1, pp.1--58, 1992.
- [GUL91] E. Gulski. Computer-Aided Recognition of Partial Discharges using Statistical Tools. *Thèse de doctorat*, Université de Delft, 1991.
- [KOH96] R. Kohavi, D. H. Wolpert. Bias plus variance decomposition for zero-one loss functions. *Machine Learning : Proceedings of the Thirteenth International Conference*, Morgan Kauffmann, pp. 275--283, 1996.
- [KON95] E. B. Kong, T. G. Dietterich. Error-correcting output coding corrects bias and variance. *Machine Learning : Proceedings of the Twelfth International Conference*, Morgan Kauffmann, pp. 313--321, 1995.
- [KRI87] T. Krishnan, S. C. Nardy. Discriminant analysis with a stochastic supervisor. *Pattern Recognition*, Vol. 20, pp. 379--384, 1987.
- [LeC86] L. Le Cun. Asymptotics Methods in Statistical Decision Theory. *Springer Series in Statistics*, Springer Verlag, 1986.
- [McL88] G. McLachlan, K. Basford. Mixture Models. *Marcel Dekker*, New York, 1988.
- [McL92] McLachlan G.J.: Discriminant Analysis and Statistical Pattern Recognition. *Wiley*, New-York, 1992.
- [McL97] G. McLachlan, T. Krishnan. The EM Algorithm and Extensions. *Wiley*, New York, 1997.
- [MIT97] T. M. Mitchell. Machine Learning. *McGraw-Hill*, New York, 1997.
- [NIG99] K. Nigam, A.K. McCallum, S. Thurn, T. Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*. pp. 1--34, 1999.

- [RED84] R. A. Redner, H. F. Walker. Mixture Densities, Maximum Likelihood and the EM Algorithm. *SIAM Review*, Vol. 26, pp. 195--239, 1984.
- [RIP96] B. D. Ripley. Pattern Recognition for Neural Networks. *Cambridge University*, 1996.
- [SAT95] L. Satich, W.S. Zaengl. Can Fractal Features be Used for Recognizing 3-d Partial Discharge Pattern? *IEEE Trans. On Electrical Insulation*, Vol. 2, pp. 352--359, 1995.
- [SCH90] R. E. Schapire. The strength of weak learnability. *Machine Learning*, N° 5, Vol. 2, pp. 197--227, 1990.
- [SCH97] R. E. Schapire, Y. Freund, P. Bartlett, W. S. Lee. Boosting the Margin : A new explanation for the effectiveness of voting methods. *Machine Learning : Proceedings of the Fourteenth International Conference*, Morgan Kauffmann, pp. 322--330, 1997.
- [SCH00] R. E. Schapire, Y. Singer. BoosTexter : A boosting-based system for text classification. *Machine Learning*, Vol. 39, 2000.
- [STO78] M. Stone. Cross-validators choice and assessment of statistical predictions. *Journal of the Royal Statistics Society*, B 36 , pp. 111--147.
- [TIT85] D. Titterington, A. Smith, U. Makov. Statistical Analysis of Finite Mixture Distributions. *Wiley Series in Probability and Mathematical Statistics*. Wiley, 1985.
- [TIT87] D. Titterington. An alternative stochastic supervisor in discriminant analysis. *Pattern Recognition*, Vol. 22, N° 1, pp. 91--95, 1987.
- [TSY73] Y. Tsyppkin. Foundations of the Theory of Learning Systems. *Mathematics in science and engineering*. Academic Press, Vol. 101, 1973.
- [WAH75] G. Wahba, S. Wold. A completely automatic French curve : fitting spline functions by cross-validation. *Communications in Statistics*, A 4, pp.1--17.
- [WAT64] G.S. Watson. A note on maximum likelihood. *Sankhya*, Vol. A, N° 26, pp. 303--304.
- [WEB99] A. Webb. Statistical Pattern Recognition. *Oxford University Press*. 1999.



# Chapitre 3

## Apprentissage avec des exemples étiquetés et non étiquetés

**Résumé.** Dans ce chapitre nous allons nous intéresser à un autre type d'apprentissage appelé *apprentissage semi-supervisé* et qui consiste à faire de l'apprentissage supervisé dans le cas où on dispose aussi d'exemples non étiquetés. Nous allons illustrer les principaux algorithmes de la littérature pour chacune des deux familles de modèles que nous avons rencontrés jusqu'ici à savoir les modèles génératifs et les modèles discriminants.

**Mot clés :** Apprentissage semi-supervisé, modèle génératif, modèle discriminant.

### 3.1 Introduction

La constitution de bases de données cohérentes et conséquentes est un des problèmes majeurs dans tous les domaines qui utilisent des techniques d'apprentissage au sens large du terme. La création de ces bases demande souvent des efforts importants dans le cadre de consortiums internationaux. Une partie importante de ce travail est souvent consacrée à l'étiquetage des données, qui peut intervenir à différents niveaux de précision en fonction de la tâche réalisée. Aujourd'hui, la multiplication des sources et des flux d'information, et les nouveaux besoins qui en résultent, changent considérablement la problématique de l'exploitation des données.

Par exemple, dans le domaine de la recherche d'information, de nombreux problèmes nécessitent des traitements massifs de données en-ligne, qui puissent s'adapter en continu à l'évolution du contenu des flux observés, et qui puissent être appliqués à des données éventuellement très différentes dans leur nature ou leur contenu. Les systèmes de traitement automatique doivent être capables de s'adapter à ces différentes contraintes, l'apprentissage vise à fournir les mécanismes permettant cette adaptation.

Pour ces problèmes, il est peu envisageable pour des questions de temps ou de ressources de procéder au développement fastidieux de bases de données étiquetées. Il en est de même dans de nombreux autres domaines qui génèrent de grandes masses de données comme la biologie par exemple.

En partant du constat que les données étiquetées sont chères alors que les données non étiquetées sont foison et que ces dernières contiennent de l'information sur le problème que l'on cherche à résoudre, la communauté apprentissage s'est récemment penchée sur le concept d'apprentissage semi-supervisé pour des tâches de discrimination et de modélisation. Il s'agit d'employer une petite quantité de données étiquetées, simultanément à une grande quantité de données non étiquetées, pour apprendre. Quelques approches ont été proposées, mais le domaine est encore très ouvert.

Un exemple pertinent pour l'apprentissage semi-supervisé est la discrimination de pages *web*. Pour employer un algorithme d'apprentissage, il faut disposer d'un ensemble de pages étiquetées à l'avance. Lang [LAN95] ont montré expérimentalement sur la base Reuters que pour un problème multi-classes, il faut avoir 1000 articles étiquetés pour espérer dépasser une précision de 50% sur seulement 10% des documents ayant les meilleurs scores. Par conséquent, il est souhaitable que l'algorithme d'apprentissage soit capable d'extraire de l'information pertinente pour la tâche à partir d'exemples non étiquetés.

Un autre exemple est le résumé de texte vu comme un problème de classification de phrases qui sera traité dans le chapitre 7. En terme d'apprentissage supervisé, pour construire un résumé de document en sélectionnant les phrases pertinentes d'un document par rapport à une requête, il faudrait disposer d'une base de données de documents étiquetés au niveau phrase, ce qui est prohibitif. Il faut donc là aussi extraire l'information pertinente des données non étiquetées.

Ce chapitre se compose de trois parties, la section 3.2 introduit quelques notations. Nous allons ensuite distinguer deux familles de modèles d'apprentissage, à savoir les modèles génératifs et discriminants. Nous allons voir les possibilités, dans chaque cas, pour apprendre en disposant d'un ensemble restreint de données étiquetées et de données non étiquetées. Les modèles discriminants seront traités en section 3.3 et les modèles génératifs en section 3.4.

### 3.2 Apprentissage semi-supervisé par la discrimination

En apprentissage supervisé les étiquettes constituent une connaissance sur le problème qui est mise à profit pendant l'apprentissage pour estimer des probabilités ou des densités liées à la distribution de probabilité jointe inconnue  $p(x,t)$ . Il existe deux grandes classes de méthodes de discrimination. La première classe est constituée par les méthodes discriminantes, celles ci vont essayer d'estimer directement la probabilité conditionnelle  $p(t/x)$ . Elles font en général peu d'hypothèses sur la nature des données ou alors des hypothèses faibles comme dans le cas de la régression logistique qui sera vue au chapitre 7. La seconde classe est celle des méthodes génératives qui vont estimer les densités  $p(x/t)$ . Ces méthodes vont souvent faire des hypothèses sur la nature des données et donc sur la forme des densités.

L'apprentissage supervisé aura un but précis (e.g. optimiser l'espérance de l'erreur de classification) qui sera traduit formellement dans un critère analytique qui en sera souvent une approximation et que l'on cherchera à optimiser.

En apprentissage non-supervisé l'information « *étiquette* » n'est plus présente, et le but de l'apprentissage est souvent beaucoup plus vague. On essaie en général de faire ressortir des régularités à partir des données avec peu d'information sur la nature des régularités. Pour cela, on va essayer, la plupart du temps, de modéliser la densité des données  $p(x)$ , en faisant différentes hypothèses sur la forme de cette densité ou de ses composantes. Ces hypothèses peuvent être vues comme des connaissances *a priori* sur le problème ou sur les données, elles sont beaucoup plus souvent des connaissances par défaut : quand on ne sait rien, on emploie les modèles classiques et simples de la littérature. Quand on utilise des métriques au lieu de densités, la forme de la métrique va jouer à cet égard le même rôle que celle de la densité.

L'apprentissage semi-supervisé est un apprentissage supervisé où l'on dispose à la fois d'exemples étiquetés et d'exemples non étiquetés. Les exemples étiquetés sont en général supposés trop peu nombreux pour que l'on puisse obtenir une bonne estimation des dépendances recherchées et l'on veut s'aider des exemples non étiquetés pour obtenir une meilleure estimation.

Pour cela, nous supposons disponibles, un ensemble d'exemples étiquetés  $D_l = \{(x_i, t_i) / i = 1, \dots, n\}$  issus de la distribution jointe  $p(x, t)$  et un ensemble d'exemples non étiquetés  $D_u = \{x_i / i = n+1, \dots, n+m\}$  supposés issus de la distribution marginale  $p(x)$ .

Si  $D_u$  est vide, on retombe sur le problème de l'apprentissage supervisé. Si  $D_l$  est vide on est en présence d'un problème d'apprentissage non-supervisé. L'intérêt de l'apprentissage semi-supervisé concerne le cas où  $m = |D_u| \gg n = |D_l|$ .

### 3.3 Algorithmes non-supervisés et semi-supervisés avec des modèles discriminants

Les idées ou méthodes employées dans de nombreux algorithmes semi-supervisés présentent des points communs avec des idées développées en non-supervisé. Aussi nous allons présenter dans la suite à la fois des techniques non-supervisées et semi-supervisées. Nous verrons ainsi en quoi les idées à la base de ces algorithmes sont semblables, même si les relations formelles restent encore à éclaircir.

- Nous allons d'abord présenter deux algorithmes non-supervisés - *décision dirigée* et *Auto-supervision* - qui apprennent à prédire des étiquettes de classe ou de groupe associées aux formes d'entrée en utilisant les sorties qu'ils prédisent pour construire des sorties désirées. Nous sommes véritablement dans un contexte non-supervisé, le but est d'attribuer les formes à des groupes sans autre information que l'observation de ces formes, mais cette « astuce » sur l'utilisation des sorties prédites permet d'avoir recours à des techniques algorithmiques similaires à celles utilisées en supervisé.

L'algorithme *décision dirigée*, utilise les exemples d'entrée pour les étiqueter avec un seul modèle. L'algorithme d'auto-supervision s'appuie sur une information représentée sous deux formes différentes et utilise deux modèles qui coopèrent dans cette classification.

- Nous allons ensuite présenter deux algorithmes semi-supervisés, le *Co-Boosting* et le *Co-Training*, qui s'appuient au départ sur un ensemble d'exemples étiquetés et qui comme l'algorithme d'auto-supervision entraînent ensuite chacun deux modèles, chaque modèle jouant alternativement le rôle de maître pour l'autre.

### 3.3.1 Décision Dirigée

La technique dite de *décision dirigée* a été mise au point dans le cadre du traitement du signal adaptatif et elle est devenue une méthode de base dans ce domaine avec de très nombreuses applications comme le filtrage, l'élimination d'interférences, l'égalisation, etc, [WID85]. De très nombreux travaux ont porté sur cette approche qui reste aujourd'hui une technique de référence. Elle a été également employée à une échelle beaucoup plus restreinte en reconnaissance des formes [DuH73]. L'idée qui est à l'origine de cette méthode est simple : dans un contexte non-supervisé, on va utiliser les sorties prédites par le système lui-même pour construire des sorties désirées. Celles-ci seront ensuite utilisées pour apprendre à partir d'une technique supervisée.

Cette approche de base a fait l'objet de nombreuses variations, en particulier :

- Elle peut être appliquée séquentiellement en mettant à jour le classifieur à chaque fois qu'une forme non étiquetée est classée (cas adaptatif ou *en-ligne*).
- Elle peut être appliquée en parallèle en attendant que tous les exemples soient classés avant de mettre à jour les paramètres du classifieur (cas *hors-ligne*).

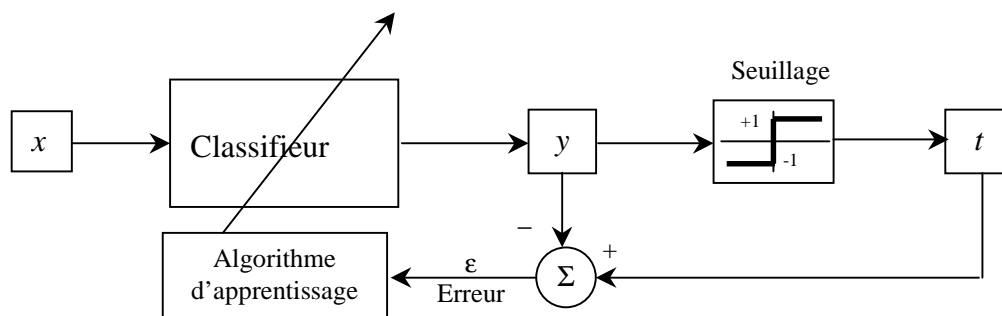


Figure 18. Mécanisme de l'algorithme de décision dirigée dans le cas adaptatif. Lorsqu'on est en présence d'un exemple non étiqueté  $x$ , le système va lui attribuer une étiquette  $t$ , en seuillant la sortie calculée pour cet exemple. Dans la deuxième étape il va apprendre la probabilité a posteriori des classes  $p(t|x)$ .

Ce processus peut aussi se répéter jusqu'à ce qu'il n'y ait plus de changement sur l'étiquetage des exemples. La figure 18, montre cet algorithme dans le cas en-ligne pour un problème de classification à deux classes.

En traitement adaptatif du signal le problème classiquement étudié dans ce cadre est un problème de classification à deux classes ; présence ou absence d'un signal dans un

environnement bruité [SCU65, SPR66, PAT66, FRA67, PAT70, AGR70]. Les vecteurs d'entrées au classifieur sont généralement représentés sous la forme [PAT66] :

$$x_i = t_i \cdot \Omega + b_i \quad (3.1)$$

Où  $\Omega$  est le signal inconnu à détecter et  $b_i$  est un vecteur de bruit additif. Le problème revient donc à donner l'étiquette  $t_i = 1$  à l'exemple  $x_i$  s'il est porteur du signal, 0 sinon. Nous nous appuyons sur la présentation de [AGR70] pour décrire un algorithme typique pour cela.

En partant d'un ensemble de formes  $X=(x_1, \dots, x_N)$  issu d'un mélange de densités :

$$p(x) = \pi_1 \cdot p(x/C_1) + \pi_2 \cdot p(x/C_2)$$

où l'une des composantes correspond au signal et l'autre au bruit, il décrit un algorithme itératif pour déterminer les paramètres  $\theta$  du modèle. Cet algorithme est le suivant, (algorithme 12).

**Entrée** : une séquence de formes  $X_m = (x_1, \dots, x_m)$ ,

Initialisation des paramètres  $\theta^{(0)}$  l'ensemble des formes  $X_0 = \{x_1\}$  et l'ensemble des étiquettes  $D_u^{(0)} = \{t_1\}$ , où  $t_1$  est une étiquette aléatoire de  $x_1$ .

Pour  $j = 1$  à  $m - 1$  faire  
{

- ① Donner une estimation de la probabilité a posteriori de la classe d'une nouvelle forme  $x_{j+1}$  ( $x_{j+1} \in X_m$  et  $x_{j+1} \notin X_j$ ) sachant les étiquettes  $D_u^{(j)}$  de l'ensemble des  $j$  formes précédentes  $X_j$  :

$$p(t_{j+1}=1 / x_{j+1}, X_j, D_u^{(j)}, \theta^{(j)})$$

Mettre à jour les nouveaux ensembles  $D_u^{(j+1)} = D_u^{(j)} \cup \{t_{j+1}\}$  et  $X_{j+1} = X_j \cup \{x_{j+1}\}$

- ② Mettre à jour les paramètres  $\theta^{(j+1)}$  du système, sachant  $X_{j+1}$  et  $D_u^{(j+1)}$  :

$$p(\theta^{(j+1)} / X_{j+1}, D_u^{(j+1)})$$

}

**Sortie** : Les paramètres du modèle décision dirigée

#### Algorithme 12. Algorithme *décision dirigée*.

On remarque que les deux étapes de cet algorithme ressemblent à l'algorithme CEM (chapitre 2, section 2.3.4 - l'étape ① est similaire aux deux étapes **E** et **C** et ② est proche de l'étape **M**). Mais ces relations n'ont pas été exploitées à notre connaissance.

De nombreux travaux, par exemple [SCU65, PAT66], étudient la convergence de la probabilité d'erreur de l'algorithme de décision dirigée dans le cas adaptatif bi-classe. Ils démontrent que cette probabilité reste bornée dans le cas où le rapport bruit/signal est peu élevé. En pratique, le comportement de l'algorithme se dégrade quand ce rapport n'est pas faible.

### 3.3.2 Auto Supervision

De Sa [DeS93a, DeS93b, DeS94a] présente un algorithme auto-supervisé pour faire de la discrimination. Cet algorithme utilise ses propres sorties pour apprendre. L'idée, qui est à la base du modèle, est d'utiliser deux classifieurs qui opèrent chacun à partir de représentations ou modalités différentes des formes que l'on veut classer, par exemple du signal sonore et de l'image. Les deux classifieurs joueront alternativement le rôle de maître et d'élève (figure 19) dans un algorithme d'apprentissage itératif : la sortie calculée par l'un sera prise comme sortie désirée par l'autre et réciproquement, jusqu'à convergence éventuelle.

Nous retrouverons cette idée sous une forme très similaire dans les algorithmes de co-training (section 3.4.5).

Le modèle fait implicitement l'hypothèse qu'il existe un lien, une "corrélacion", entre les 2 modalités et que la procédure d'apprentissage doit faire émerger ce lien en se basant sur la "cohérence" des réponses des deux classifieurs.

Cette idée a été appliquée à l'apprentissage des vecteurs de référence dans un classifieur linéaire par morceaux du type plus proches voisins. De Sa a montré comment son algorithme optimisait la cohérence des réponses des classifieurs. Ci-dessous, Nous présentons brièvement ces travaux.

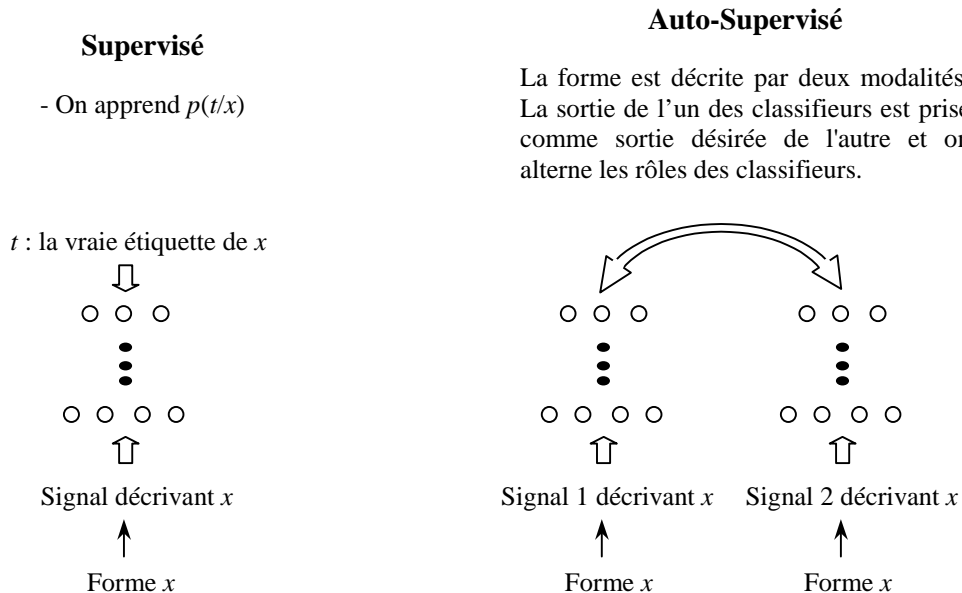


Figure 19. apprentissage supervisé vs apprentissage auto-supervisé

### 3.3.2.1 Etiquetage et apprentissage de vecteurs de référence

[DeS94a] considère un classifieur linéaire par morceaux, basé sur le calcul de distance par rapport à des prototypes. Elle propose un algorithme auto-supervisé dont les principales étapes sont les suivantes :

- Initialisation et étiquetage des vecteurs prototypes.
- Auto-apprentissage des vecteurs prototypes.

Pour la première étape elle propose d'initialiser les prototypes par un algorithme non-supervisé pour chaque modalité, puis de les étiqueter par auto-supervision. La figure 20 décrit le système d'apprentissage qui est composé d'une première couche de neurones dont les poids sont les prototypes, la couche cachée pour chaque modalité est du type "winner takes all" et seul le vecteur prototype le plus proche de la forme d'entrée aura une sortie à 1, les autres étant à 0. La seconde couche de poids calcule une somme pondérée de ses entrées. Les vecteurs de poids de la couche cachée liant la sortie à 1 de chaque modalité vers le neurone de sortie le plus actif seront augmentés. L'algorithme permet un étiquetage des vecteurs de référence. Ceux-ci peuvent être appris dans un second temps<sup>10</sup>:

- Pour chaque modalité, la classe prédite de l'exemple est celle du vecteur de référence qui lui est le plus proche.
- Si les deux classifieurs ne sont pas en accord, alors les vecteurs de poids sont modifiés suivant une règle d'apprentissage heuristique de façon à réduire ce désaccord.

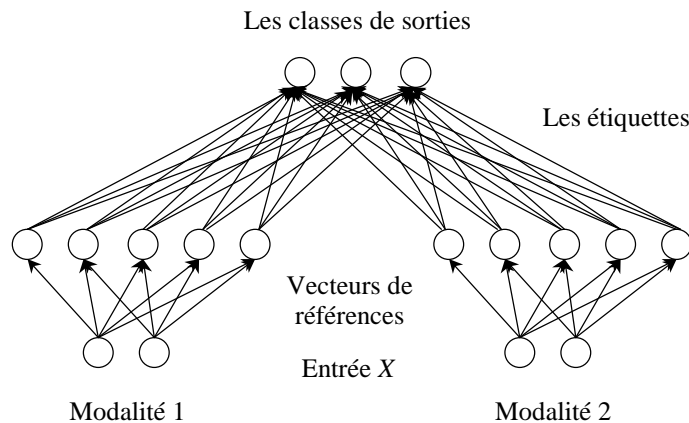


Figure 20. Le réseau pour l'apprentissage des étiquettes des vecteurs de référence. Les vecteurs poids des neurones de la couche cachée représentent les vecteurs de références et les vecteurs poids connectant les neurones de la couche cachée et les neurones de sorties représentent les classes. Dans ce schéma il y a 3 classes et deux modalités.

<sup>10</sup> Nous ne détaillons pas l'algorithme, qui utilise une formulation très particulier propre à l'algorithme LVQ2 [KOH90].

### 3.3.2.2 Optimisation de la cohérence

En dehors de l'algorithme lui-même, le résultat suivant nous a paru intéressant. [DeS94b] montre que son algorithme permet d'optimiser la cohérence des deux classifieurs. Il considère un problème à 2 classes, chacune avec un seul mode.

Notons  $b_1$  et  $b_2$  respectivement les frontières de décision entre les classes pour chaque modalité (figure 21).

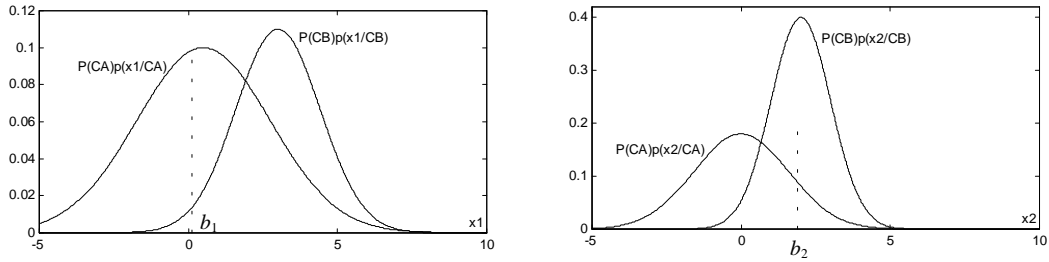


Figure 21. Si une modalité reçoit un exemple de sa distribution de classe A, la modalité 2 reçoit un exemple de sa propre distribution de classe A. Sans aucune information sur la classe des exemples, les deux classifieurs doivent déterminer le placement des frontières  $b_1$  et  $b_2$  appropriées.

On peut écrire analytiquement l'erreur de désaccord comme :

$$E(b_1, b_2) = Pr\{x_1 < b_1 \ \& \ x_2 > b_2\} + Pr\{x_1 > b_1 \ \& \ x_2 < b_2\}$$

Où  $x_1$  et  $x_2$  décrivent la même forme  $X$  pour chaque modalité.

Cette équation est indépendante des classes et elle peut s'écrire sous la forme générale :

$$E(b_1, b_2) = \int_{-\infty}^{b_1} \int_{b_2}^{\infty} f(x_1, x_2) dx_1 dx_2 + \int_{b_1}^{\infty} \int_{-\infty}^{b_2} f(x_1, x_2) dx_1 dx_2$$

où

$$f(x_1, x_2) = p(C_A)p(x_1 / C_A)p(x_2 / C_A) + p(C_B)p(x_1 / C_B)p(x_2 / C_B)$$

$f(x_1, x_2)$  est la densité de probabilité jointe des deux modalités. Ce qui fournit une interprétation du but optimisé par cet algorithme.

L'algorithme a été testé sur des problèmes relativement simples, il n'est sûrement pas très performant. Toutefois, les idées proposées sont intéressantes et ont été reprises dans des algorithmes plus sophistiqués par la suite.

### 3.3.4 Co-Boosting

Cet algorithme est une extension des algorithmes de Boosting (cf. 2.2.6.2) dans le cas où l'on dispose d'exemples non étiquetés en plus d'exemples étiquetés [COL99]. Comme nous l'avons vu, l'algorithme AdaBoost est un algorithme supervisé qui trouve une combinaison pondérée d'un ensemble de classifieurs construits à partir d'un même modèle mais avec des échantillonnages ou des pondérations différentes des exemples. Les échantillonnages successifs et les poids de la combinaison sont choisis de façon à minimiser l'erreur de classification. L'algorithme de Co-Boosting présente une approche similaire, mais il s'attache à minimiser une erreur de désaccord, comme dans



le cas de l'algorithme auto-supervisé de De Sa. Cet algorithme construit itérativement deux classifieurs : à chaque itération on s'attache à minimiser une fonction continue différentielle qui représente le nombre d'exemples auxquels les deux classifieurs donnent des étiquettes différentes.

Plus précisément, chaque forme  $x_i$  est décrite par une paire  $(x_{1,i}, x_{2,i})$ . Les  $x_{k,i}$  représentent les 2 modalités caractérisant l'exemple  $x_i$ . On supposera là aussi que les  $n$  premières paires ont des étiquettes  $t_i \in \{-1, +1\}$  tandis que les  $m$  paires suivantes sont non étiquetées. De plus, Collins et Singer [COL99] font l'hypothèse que chacune des modalités  $x_{1,i}$  et  $x_{2,i}$  est suffisante pour déterminer l'étiquette  $t_i$ . La tâche d'apprentissage est de trouver deux classifieurs  $g_1$  et  $g_2$  tel que

- Pour tout  $(x_i, t_i)$  appartenant à l'ensemble des exemples étiquetés  $D_l$ ,  $\text{sgn}(g_1(x_{1,i})) = \text{sgn}(g_2(x_{2,i})) = t_i$ ,
- Et pour tous exemples  $x_i$  de l'ensemble non étiquetés  $D_u$ ,  $\text{sgn}(g_1(x_{1,i})) = \text{sgn}(g_2(x_{2,i}))$ .

Comme dans Adaboost, les deux classifieurs  $g_1$  et  $g_2$  seront construits comme des combinaisons linéaires de classifieurs de base  $h_1$  et  $h_2$ . Ainsi, à chaque itération  $j$ , la mise à jour de ces classifieurs est :

$$\forall i, k = 1, 2 \quad g_k^{(j+1)}(x_{k,i}) = g_k^{(j)}(x_{k,i}) + \alpha_j \cdot h_k^{(j)} \quad (3.2)$$

Où  $\alpha_j$  est le coefficient du classifieur définit plus bas. Le but de cet algorithme est de minimiser une borne de l'erreur en apprentissage qui est inspirée de celle proposée par Freund et Shapire dans le cas du boosting [FRE96]. Cette mesure a la forme suivante :

$$\begin{aligned} Z_{co} = & \sum_{i=1}^n [\exp(-t_i g_1(x_{1,i})) + \exp(-t_i g_2(x_{2,i}))] \\ & + \sum_{i=n+1}^{n+m} [\exp(-\text{sgn}(g_2(x_{2,i}))g_1(x_{1,i})) + \exp(-\text{sgn}(g_1(x_{1,i}))g_2(x_{2,i}))] \end{aligned} \quad (3.3)$$

Si  $Z_{co}$  est petit, il en découle que les deux classifieurs ont un petit taux d'erreur sur les exemples étiquetés et qu'ils donnent aussi la même étiquette sur un grand nombre d'exemples non étiquetés. Le premier terme vise à minimiser l'erreur de classification sur les exemples étiquetés, et le second terme à minimiser le désaccord.  $Z_{co}$  est une borne supérieure de l'erreur de classification sur les exemples étiquetés et de l'erreur de désaccord sur les exemples non étiquetés. L'algorithme 13 décrit l'algorithme de Co-Boosting.

A chaque itération, l'algorithme de Co-Boosting met à jour les 2 classifieurs : chacun apprend alternativement alors que l'autre est fixé. Soient  $\{g_k^{(j-1)}\}_{k=1,2}$  les deux classifieurs obtenus à l'itération  $j-1$  et supposons que c'est le tour du classifieur 1 d'être mis à jour tandis que le classifieur 2 est gardé fixe. [COL99] définit des pseudo-étiquettes:

$$\tilde{t}_i = \begin{cases} t_i & 1 \leq i \leq n \\ \text{sgn}(g_2^{(j-1)}(x_{2,i})) & n+1 \leq i \leq n+m \end{cases}$$

**Entrée :** Une base d'apprentissage  $S = \{(x_{1,i}, x_{2,i})\}_{i=1,\dots,n+m}$ ,  $\{t_i\}_{i=1,\dots,n}$  un entier  $\mathbf{I}$  (nombre d'itérations).

Initialiser  $\forall i, k : g_k^{(0)}(x_i)$

Pour  $j = 1$  à  $\mathbf{I}$  et pour  $k = 1, 2$  faire

{

- Mettre à jour les pseudo-étiquettes

$$\tilde{t}_i = \begin{cases} t_i & 1 \leq i \leq n \\ \text{sgn}(g_{3-k}^{(j-1)}(x_{3-k,i})) & n+1 \leq i \leq n+m \end{cases}$$

- Mettre à jour la distribution virtuelle

$$Dis_k^{(j)}(i) = \frac{1}{Z_k^{(j)}} \exp(-\tilde{t}_i \cdot g_k^{(j-1)}(x_{k,i}))$$

Où  $Z_k^{(j)}$  est un facteur de normalisation.

- Entraîner le classifieur de base  $h_k^{(j)}$  sur la base obtenue d'après la distribution  $Dis_k^{(j)}$

- Calculer le poids  $\alpha_j$  du classifieur  $h_k^{(j)}$

$$\alpha_j = \frac{1}{2} \ln \left( \frac{W_+}{W_-} \right) \text{ avec } W_+ = \sum_{i/h_1^{(j)}(x_{1,i})=\tilde{t}_i} Dis_1^{(j)}(i) \text{ et } W_- = \sum_{i/h_1^{(j)}(x_{1,i})=-\tilde{t}_i} Dis_1^{(j)}(i).$$

- Mettre à jour les classifieurs globaux qui combinent les classifieurs de base à l'étape  $j$  :

$$\forall i, g_k^{(j)}(x_{k,i}) = g_k^{(j-1)}(x_{k,i}) + \alpha_j h_k^{(j)}(x_{k,i})$$

}

$$C^*(x) = \text{sgn} \left( \sum_{k=1}^2 g_k^{(\mathbf{I})}(x) \right)$$

**Sortie :** classifieur  $C^*$

---

Algorithme 13. L'algorithme de Co-Boosting

Les exemples étiquetés gardent leur étiquette tandis que le second classifieur fournit celle des exemples non étiquetés. Une nouvelle distribution pour la modalité 1 est calculée par :

$$Dis_1^{(j)}(i) = \frac{1}{Z_1^{(j)}} \exp(-\tilde{t}_i \cdot g_1^{(j-1)}(x_{1,i}))$$

Où  $Z_1^{(j)}$  est le terme de normalisation pour que  $D_1^{(j)}$  soit une distribution. On construit alors une nouvelle base d'apprentissage en échantillonnant d'après cette nouvelle distribution, et on entraîne un classifieur de base  $h_1^{(j)}$  sur cette base. Le poids  $\alpha_k$  de ce classifieur est calculé comme :

$$\alpha_j = \frac{1}{2} \log \left( \frac{W_+}{W_-} \right)$$

$$\text{où } W_+ = \sum_{i/h_1^{(j)}(x_{1,i})=\tilde{t}_i} Dis_1^{(j)}(i) \text{ et } W_- = \sum_{i/h_1^{(j)}(x_{1,i})=-\tilde{t}_i} Dis_1^{(j)}(i).$$

Cette procédure est répétée **I** fois en alternant les deux classifieurs.

Cet algorithme a été utilisé sous une forme légèrement modifiée pour la classification d'entités nommées : dans leur exemple il s'agit d'associer à des séquences déjà étiquetées comme étant des noms propres une des trois catégories - Personne, Entreprise, Lieu. L'algorithme de base utilisé, est basé sur les listes de décision. [COL99] compare différents algorithmes semi-supervisé et montrent, entre autre, que les méthodes discriminantes comme Co-Boosting ou des variantes de Co-Training (voir ci-dessous) obtiennent sur ce problème des performances bien supérieures aux méthodes génératives.

### 3.3.5 Co-Training

Blum et Mitchell [BLU98] ont introduit l'algorithme de Co-Training qui repose sur des idées assez similaires à l'algorithme d'auto-supervision de De Sa ou à l'algorithme de Co-Boosting. Là aussi, dans un contexte semi-supervisé, on suppose que l'on dispose de deux modalités pour décrire chaque exemple, et que chaque description est assez riche pour apprendre les paramètres des deux classifieurs dans le cas où on aurait disposé d'assez d'exemples étiquetés. On va ensuite entraîner les classifieurs en leur faisant alternativement jouer le rôle de maître et d'élève. Les deux classifieurs sont d'abord entraînés séparément sur les données étiquetées. On tire ensuite aléatoirement des exemples de  $D_u$  qui sont étiquetés par chacun des deux classifieurs, la sortie calculée par le classifieur 1 sert de sortie désirée pour le classifieur 2 et réciproquement. L'algorithme est détaillé dans 14, chaque itération dans cet algorithme, seuls quelques exemples supplémentaires sont étiquetés. Bien qu'il ne soit pas formulé comme tel, on se retrouve alors avec un algorithme d'optimisation stochastique qui est assez proche des algorithmes adaptatifs dans son principe de base. La mise en œuvre qui est proposée est assez simple et s'apparente à l'algorithme de décision dirigée avec la différence que l'on a ici 2 classifieurs, chacun fournissant alternativement les étiquettes correspondant

aux sorties désirées de l'autre classifieur. L'originalité du travail consiste en une analyse du Co-Training dans un cadre PAC (*Probably Approximately Correct*) qui permet aux auteurs d'étudier l'apprenabilité de concepts par Co-Training.

[BLU98] utilisent cet algorithme pour classifier des pages web en deux classes pertinentes/non-pertinentes par rapport à une tâche donnée.

En partant de deux représentations différentes pour chaque page, à savoir le sac de mots contenu dans chaque page et le sac de mots contenu dans les hyper-liens de ces pages, ils apprennent les paramètres d'un classifieur Naïve Bayes (décrit dans la section suivante) en s'appuyant sur chacune de ces représentations.

Là aussi, les expériences montrent que les exemples non étiquetés permettent d'augmenter sensiblement les performances des classifieurs.

---

**Entrée :** Une base étiquetée  $D_l = \{(x_{1,i}, x_{2,i}), t_i\}_{i=1,\dots,n}$ ,

Une base non étiquetée  $D_u = \{(x_{1,i}, x_{2,i})\}_{i=n+1,\dots,n+m}$ ,

Deux classifieurs  $h_1$  et  $h_2$ ,

Un nombre d'itération  $I$ ,

Créer une base  $D'_u$  en choisissant aléatoirement  $N$  exemples de  $D_u$ .

Pour  $j = 1$  à  $I$

{

- Utiliser  $D_l$  pour entraîner le classifieur  $h_1$
- Utiliser  $D_l$  pour entraîner le classifieur  $h_2$
- Etiqueter avec  $h_1$ ,  $p$  exemples positifs et  $n$  exemples négatifs de  $D'_u$
- Etiqueter avec  $h_2$ ,  $p$  exemples positifs et  $n$  exemples négatifs de  $D'_u$
- Ajouter ces exemples auto-étiquetés à  $D_l$
- Choisir aléatoirement  $2p+2n$  exemples de  $D_u$  et les ajouter à  $D'_u$

}

**Sortie :** classifieur  $C^* = \text{Combinaison}(h_1, h_2)$ .

---

Algorithme 14. L'algorithme de Co-Training

### 3.4 Algorithmes semi-supervisé avec un modèle génératif

Dans cette classe de méthodes on part d'un modèle génératif pour estimer la densité  $p(x)$ . Une des approches qui a été proposée par plusieurs auteurs consiste à se placer dans le cadre du maximum de vraisemblance et à considérer un modèle de mélange pour  $p(x)$ , (cf section 2.3.2), dont les composantes sont des distributions simples e.g. des gaussiennes. Les étiquettes inconnues de  $D_u$  sont alors considérées comme des données manquantes et l'algorithme EM est employé pour estimer ces étiquettes et en déduire les

densités composantes  $p(x/t)$ . Après estimation, la règle de Bayes permettra de calculer les probabilités a posteriori  $p(t/x)$ . Il s'agit d'une application directe et naturelle de l'algorithme EM pour maximiser une vraisemblance. Cette approche générale est décrite avec suffisamment de détails dans [McL88a, McL88b] ainsi que son application dans le cas d'un mélange de gaussienne qui a l'avantage de fournir une solution analytique au problème. Cette méthode de base a été redécouverte et exploitée sous différentes variantes par de nombreux auteurs, qui n'ont rien apporté de supplémentaire au niveau des idées ou de l'analyse mais qui ont quand même souvent le mérite d'avoir réalisé des tests assez importants et d'avoir introduit des variantes algorithmiques robustes. Dans la suite, nous nous appuyerons sur les travaux de Nigam et de ses collaborateurs qui ont utilisé des classifieurs de base très simples et qui ont appliqué leurs développements à la classification de textes.

### 3.4.1 Maximiser la vraisemblance d'un mélange

Nigam et al. [NIG99] utilisent l'approche précédente pour apprendre à classifier des textes. Les données - les représentations des textes - sont supposées générées par un mélange de densités dont les composantes sont des classifieurs naïve Bayes. Le critère d'apprentissage est la vraisemblance des données. L'algorithme d'optimisation utilisé est EM. L'algorithme d'apprentissage initialise d'abord le classifieur en utilisant les documents étiquetés. Il étiquette ensuite les documents non étiquetés à partir de ce classifieur de base. Il entraîne après un nouveau classifieur en utilisant toutes les étiquettes des documents et réitère ce processus jusqu'à la convergence.

Pour illustrer le fonctionnement de ce type de méthodes, nous décrivons ci-dessous l'algorithme naïve Bayes spécifique utilisé par les auteurs et l'algorithme semi-supervisé.

#### 3.4.1.1 Le classifieur Naïve Bayes

La dénomination Naïve Bayes recouvre une famille de classifieurs qui traitent des données de taille fixe (vecteurs) ou des séquences, qui peuvent avoir des formes différentes, mais font tous l'hypothèse que les termes des vecteurs sont indépendants. Nous décrivons l'algorithme dans le contexte de la classification de documents tel qu'il a été proposé par [NIG99].

Supposons qu'un document  $x_i$  soit décrit par un vecteur de termes  $\langle m_1^i, m_2^i, \dots, m_{|x_i|}^i \rangle$  où chaque mot provient d'un dictionnaire  $D = \langle m_1, m_2, \dots, m_{|D|} \rangle$ . Le modèle probabiliste utilisé est un modèle de mélange :

$$p(x_i / \theta) = \sum_{k=1}^c \pi_k p(x_i / C_k = t_i, \theta)$$

Où  $\theta$  désigne les paramètres du modèle et  $c$  représente le nombre total de classes. Les composantes du mélange sont des classifieurs "naïve bayes" :

$$p(x_i / C_k = t_i, \theta) = p(|x_i|) \prod_{j=1}^{|x_i|} p(m_j^i / C_k, \theta) \quad (3.2)$$

Dans cette équation, les termes du produit  $p(m_t / C_k = t_i, \theta)$ ,  $t = \{1, \dots, |D|\}$  sont les probabilités conditionnelles des mots du document dans la classe : ils vérifient  $\sum_t p(m_t / C_k = t_i, \theta) = 1$  et sont estimés par comptage - c'est un modèle d'unigramme pour

chaque classe. Le terme  $p(|x_i|)$  prend en compte la longueur du document. [NIG99] font l'hypothèse que la longueur des documents est uniformément distribuée pour toutes les classes de documents, ce qui permet de ne pas tenir compte de ce paramètre longueur. La collection complète des paramètres de ce modèle correspond aux probabilités conditionnelles des unigrammes et aux probabilités a priori des classes :

$$\theta = \{ p(m_t / C_k, \theta), m_t \in D, C_k \in C, \pi_k \}$$

L'apprentissage de ce classifieur consiste alors à estimer  $\theta$  en utilisant l'ensemble des documents étiquetés,  $D_l = \{x_1, \dots, x_n\}$ .

Les estimations de probabilité des mots sont :

$$\hat{\theta}_{m_t / C_k} \equiv p(m_t / C_k, \hat{\theta}) = \frac{1 + \sum_{i=1}^n t_{ki} \cdot N(m_t, x_i)}{|D| + \sum_{t=1}^{|D|} \sum_{i=1}^n t_{ki} \cdot N(m_t, x_i)} \quad (3.3)$$

Où  $N(m_t, x_i)$  est le nombre d'occurrence du terme  $m_t$  dans le document  $x_i$  et où les  $t_{ki}$  sont les variables indicatrices de classe, i.e.  $x_i \in C_k \Leftrightarrow t_{ki} = 1$ .

Les probabilités a priori des classes  $\pi_k$  sont estimées de la même façon et elles sont données par :

$$\pi_k = \frac{1 + \sum_{i=1}^n t_{ki}}{c + n} \quad (3.4)$$

Une fois connus ces paramètres, les probabilités a posteriori des classes  $c_j$  sachant les documents  $d_i$  sont obtenues par la règle de Bayes :

$$\begin{aligned} p(t_i = C_k / x_i, \hat{\theta}) &= \frac{p(C_k / \hat{\theta}) p(x_i / C_k, \hat{\theta})}{p(x_i / \hat{\theta})} \\ &= \frac{p(C_k / \hat{\theta}) \prod_{j=1}^{|x_i|} p(m_j^i / C_k, \hat{\theta})}{\sum_{r=1}^{|C|} p(C_r / \hat{\theta}) \prod_{j=1}^{|x_i|} p(m_j^i / C_r, \hat{\theta})} \end{aligned} \quad (3.5)$$

La classe d'un document  $x_i$  est alors celle qui donne la plus grande probabilité a posteriori  $p(t_i = C_k / x_i, \hat{\theta})$ .

### 3.4.1.2 L'algorithme semi-supervisé basé sur EM

[NIG99] ont considéré les étiquettes de documents comme des variables manquantes pour l'ensemble des documents non étiquetés  $D_u$  et ils ont appliqué l'algorithme EM pour estimer les paramètres  $\theta$  du classifieur Naïve Bayes. Cet algorithme permet de trouver un optimum local de la vraisemblance des données  $D_l$  et  $D_u$  qui s'exprime sous la forme suivante :

$$p(D_l, D_u / \theta) = \prod_{x_i \in D_u} \sum_{k=1}^c \pi_k p(x_i / C_k; \theta) \prod_{x_i \in D_l} \pi_k p(x_i / C_k = t_i; \theta) \quad (3.6)$$

Le premier terme du produit est la vraisemblance pour les données non étiquetées qui obéissent à un modèle de mélange, et le second terme la vraisemblance des données étiquetées pour lesquelles la classe des documents est connue. A cause de la somme

dans le premier terme, il n'existera pas en général de solution analytique à ce problème et l'on a recours à des algorithmes itératifs, ici EM pour trouver un optimum local. L'algorithme 15 présente cette procédure. Elle est semblable à l'algorithme EM classique pour des modèles de mélange, la seule différence est dans l'Initialisation et dans la connaissance certaine des probabilités conditionnelles pour les documents étiquetés.

---

**Entrée :** Une base étiquetée  $D_l = \{x_i, t_i\}_{i=1, \dots, n}$ ,

Une base non étiquetée  $D_u = \{x_i\}_{i=n+1, \dots, n+m}$

- Initialiser le classifieur Naïve Bayes avec les documents étiquetés  $D_l$

Faire jusqu'à ce que la vraisemblance des paramètres ne varie plus

{

- Etape E : Utiliser les valeurs courantes  $\hat{\theta}$  du classifieur pour estimer la probabilité conditionnelle d'appartenance de chaque document non étiqueté dans  $D_u$ , i.e. estimer  $p(t_i/x_i, \hat{\theta})$ ,  $\forall x_i \in D_u$ . Pour cela on utilise l'équation (3.5). Pour les documents étiquetés, cette probabilité est 1 pour la bonne classe, 0 pour les autres.
- Etape M : Re-estimer les paramètres du classifieur  $\hat{\theta}$  en utilisant toutes les probabilités conditionnelles calculées à l'étape précédente. Pour cela on applique les équations (3.3) et (3.4).

}

**Sortie :** classifieur  $\hat{\theta}$  qui prédit la classe d'un document non étiqueté.

---

Algorithme 15. L'algorithme semi-supervisé basé sur l'algorithme EM

[NIG99] ont procédé à des expériences intensives pour la classification, sur différents corpus textuels, avec différentes variantes de l'algorithme, en pondérant l'importance respective des données étiquetées et non étiquetées. Ils proposent une analyse de leurs résultats expérimentaux.

### 3.5 Conclusion

Nous avons introduit dans ce chapitre le concept d'apprentissage semi-supervisé et nous avons décrit deux grandes classes de techniques qui s'appuient respectivement sur des modèles discriminants et sur des modèles génératifs. Nous avons également décrit les principaux algorithmes de la littérature pour chacune de ces catégories. Le concept d'apprentissage semi-supervisé a été développé par les statisticiens dans les années 80, puis repris plus récemment en informatique dans des contextes parfois différents (e.g. apprentissage à partir de plusieurs modalités). Nous avons montré que les algorithmes employés présentaient de nombreuses similarités avec des algorithmes non-supervisés classiques. Au-delà de cette présentation des idées et techniques employées, de nombreuses questions de fond restent ouvertes. L'apport des données non-supervisées

est en général étudié de façon purement empirique. Quelques approches plus formelles ont été proposées sous des hypothèses restrictives. Il existe sûrement un compromis sur la quantité d'information que l'on veut extraire des données non étiquetées. Supposons que l'on parte de classifieurs entraînés sur des données étiquetées, ils prédiront la classe des données non étiquetées avec une certaine confiance. Si l'on se contente de considérer les étiquettes prédites qui ont une forte confiance associée, ces données n'apporteront que peu d'information supplémentaire par rapport à la base étiquetée. Si l'on considère des étiquettes avec une faible confiance, elles risquent de ne pas être cohérentes avec les autres données. Il y a clairement un compromis à trouver entre ces deux extrêmes, mais cette voie a été peu explorée.

Enfin, les deux classes de méthodes que nous avons présentées obéissent à deux philosophies différentes : les méthodes discriminantes vont directement essayer d'estimer  $p(t/x)$  et les méthodes génératives demandent de résoudre un problème plus complexe qui est la modélisation de  $p(x)$ . L'adéquation de chacune de ces démarches au problème de l'apprentissage semi-supervisé est encore à explorer.



### 3.6 Bibliographie

- [AGR70] A. K. Agrawala. Learning with a Probabilistic Teacher. *IEEE Transactions on Information Theory*, Vol. IT-16, N° 4, pp. 373--379, 1970.
- [BLU98] A. Blum, T. Mitchell. Combining Labeled and Unlabeled Data with Co-Training. *Proceedings of the Conference on Computational Learning Theory*, pp. 92--100, 1998.
- [CEL92] G. Celeux, G. Govaert. A Classification EM algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*. Vol. 14, pp. 351--332, 1992.
- [CEL93] G. Celeux, G. Govaert. Comparison of the Mixture and the classification Maximum Likelihood in Cluster Analysis. *Journal of Statistical Computation and Simulation*. Vol. 14., pp. 127--146, 1993
- [COL99] M. Collins, Y. Singer. Unsupervised models for named entity classification. *In Proceedings of EMNLP*, 1999.
- [CRA46] H. Cramer. *Mathematical Methods of Statistics*. Princeton, p. 248, 1946.
- [DeS93a] V. De Sa. Learning classification with unlabeled data. *Neural Information Processing Systems*, Vol. 6, pp. 112--119, 1993.
- [DeS93b] V. De Sa, D. H. Ballard. A Note on Learning Vector Quantization. *Neural Information Processing Systems*, Vol. 5, pp. 220--227, 1993.
- [DeS94a] V. De Sa. Unsupervised Classification Learning from Cross-Modal Environmental Structure, *Thèse de doctorat*, 1994.
- [DeS94b] V. De Sa. Minimizing Disagreement for Self-Supervised Classification. *Proceedings of the 1993 Connectionist Summer School*, pp. 300--307, 1994.
- [DuH73] R. O. Duda, P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.
- [FRA67] S. C. Fralick. Learning to Recognize Patterns without a Teacher. *IEEE Transactions on Information Theory*, Vol. IT-13, N° 1, pp. 57--64, 1967.
- [FRE96] Y. Freund, R. E. Schapire. Experiments with a new boosting algorithm. *Machine Learning : Proceedings of the Thirteenth International Conference*, Morgan Kauffmann, pp. 148--156, 1996.
- [FLU97] B. Flury. *A First Course in Multivariate Statistics*. Springer, 1997.
- [KOH90] T. Kohonen, Statistical Pattern Recognition Revisited, *Advanced Neural Computers*, R. Eckmiller (Ed.) (Elsevier, 1990) pp. 137--144.

- [LAN95] K. Lang. Newsweeder: Learning to filter netnews. *Proceedings of the Twelfth International Conference on Machine Learning*. pp. 331--339, 1995.
- [McL88a] G.J. McLachlan, K. E. Basford. Mixture Models : Inference and Applications to Clustering. *Marcel Dekker*, New-York, 1988.
- [McL88b] G.J. McLachlan, R. D. Gordon. Fitting mixture models to grouped and truncated data via the EM algorithm. *Biometrics*, Vol. 44, pp. 571--578, 1988.
- [NIG99] K. Nigam, A.K. McCallum, S. Thurn, T. Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*. pp. 1--34, 1999.
- [PAT66] E. A. Patrick, J. C. Hancock. Nonsupervised Sequential Classification and Recognition of Patterns. *IEEE Transactions on Information Theory*, Vol. IT-12, N° 3, pp. 362--372, 1966.
- [PAT70] E. A. Patrick, J. P. Costello, F. C. Monds. Decision-Directed Estimation of a Two-Class Decision Boundary. *IEEE Transactions on Information Theory*, Vol. C-9, N° 3, pp. 197--205, 1970.
- [SCU65] H. J. Scudder. Adaptive Communication Receivers. *IEEE Transactions on Information Theory*, pp. 167--174, 1966.
- [SPR66] J. Spragins. Learning Without a Teacher. *IEEE Transactions on Information Theory*, Vol. IT-12, N° 2, pp. 223--230, 1966.
- [WID85] B. Widrow, S.D. Stearns. Adaptive Signal Processing. *Prentice-Hall*, 1985.
- [YAR95] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. *In Proceedings of the 33<sup>rd</sup> Annual meeting of the Association for Computational Linguistics*. pp. 189--196, 1995.

## **Partie II**

### **Fouille de Données Textuelles**



# Chapitre 4

## Accès à l'information textuelle

**Résumé.** Dans ce chapitre nous introduisons les techniques de base utilisées en Extraction d'Information (EI) en mettant l'accent sur les conférences MUCs qui ont permis le développement des systèmes opérationnels d'extraction actuels. Nous décrivons brièvement l'évolution des systèmes d'EI vus au travers de ces conférences et les problèmes d'évaluation. Nous présenterons ensuite une introduction aux tâches de base de la Recherche d'Information (RI), nous donnerons un aperçu des modèles et techniques classiques ainsi que les mesures d'évaluation qualitatives des systèmes de RI.

**Mots clés :** Extraction d'Information, Recherche d'Information, Evaluation.

### 4.1 Introduction

L'utilisation généralisée du *web* a causé une demande sans précédent de systèmes capables de sélectionner, de structurer, et d'extraire les informations textuelles disponibles. Les nouveaux flux d'information et les grandes bases de données qui commencent à être disponibles créent également des besoins nouveaux auxquels les différentes communautés du texte essaient de répondre en adaptant les approches qu'elles ont développées depuis de nombreuses années. Ces communautés sont actuellement en pleine évolution et les frontières traditionnelles qui avaient été dessinées au cours des années sont largement bouleversées. On assiste à l'émergence d'un domaine qui est au confluent de leurs préoccupations et que l'on appellera l'« accès à l'information textuelle ». Dans ce chapitre, nous allons introduire des problématiques classiques étudiées par deux de ces communautés qui sont celles de la Recherche d'Information (RI) et de l'Extraction d'Information (EI). Nous allons présenter les principaux outils qu'elles utilisent pour l'accès à l'information textuelle. Les travaux que nous avons développés entrent dans les problématiques étudiées en RI et EI, ceci bien que les approches employées soient un peu différentes et centrées sur l'apprentissage. Ces deux domaines de recherche existent depuis de nombreuses années et il est bien sûr hors de question de présenter de façons exhaustives les problèmes étudiés et les approches développées. Nous nous contenterons de donner une idée générale de ces problèmes et approches à l'usage du chercheur en IA non-spécialiste du texte, nous présenterons les notions de base dont nous aurons besoin dans les chapitres

ultérieurs, et les idées sur lesquelles nous nous sommes appuyés pour travailler. Nous renvoyons à des ouvrages classiques pour une présentation détaillée de ces domaines.

L'objectif initial de la RI est de retrouver une information souhaitée (des documents en général) parmi un ensemble d'informations disponibles. Une des tâches de base abordée dans ce domaine est de fournir à un utilisateur qui formule une requête une liste de documents pertinents susceptibles de répondre à cette demande. Les systèmes de RI qui permettent de traiter de grandes masses de documents se doivent d'être robustes et rapides, ils reposent très souvent sur des techniques statistiques assez simples. Cette tendance à privilégier des méthodes un peu frustrées par rapport à celles développées en traitement automatique du langage naturel a été favorisée par le développement de compétitions sur des très grands corpus comme c'est le cas dans les conférences TREC [TREC]. Traditionnellement, l'objet de base en RI est le texte pris dans sa globalité ou éventuellement des portions de texte. Ces textes seront en général représentés de façon globale sous la forme d'un vecteur de termes. On peut cependant noter que d'autres tâches ont récemment été proposées dans les compétitions TREC où il faut extraire des informations de type groupe de mots au sein des textes.

L'extraction d'information consiste à extraire d'un texte en langage naturel les informations pertinentes pour une requête prédéfinie, qui pourra correspondre à une demande relativement complexe. Il s'agit cette fois de « *comprendre* » le sens du document afin de pouvoir satisfaire à cette demande. Les systèmes d'extraction se sont développés notamment autour de la compétition MUC [MUC3, MUC4, MUC4, MUC5, MUC6, MUC7] qui s'est concentrée autour d'une spécialisation de la tâche d'extraction : le remplissage de formulaire à partir d'un texte libre. Les techniques d'EI sont principalement issues du traitement du langage naturel. Les systèmes développés comportent toujours des modules très spécifiques de la tâche étudiée. On a toutefois assisté ces dernières années à des essais de développement de systèmes génériques qui font appel à des techniques d'apprentissage pour s'adapter aux corpus traités et à la tâche d'extraction.

Parce qu'elles avaient des buts différents et qu'elles ont développé des méthodes également très différentes, les deux communautés RI et EI sont restées assez séparées et ont eu des évolutions parallèles. Avec l'évolution des demandes en accès à l'information textuelle, qui conduisent la RI à se pencher sur l'extraction d'information au sein des textes et l'EI à développer des systèmes qui soient plus génériques, les frontières des domaines ne sont plus aussi nettes.

Ce chapitre est organisé autour de ces deux thèmes, nous étudierons d'abord les techniques de bases utilisées en EI (section 2), Dans la deuxième partie nous présenterons les tâches traitées par les systèmes de RI (section 3).

## 4.2 Extraction de l'Information

L'extraction d'information nécessite une compréhension des textes qui soit suffisante pour répondre à la tâche spécifique étudiée. Elle s'est souvent focalisée sur des problématiques limitées qui visent à extraire à partir d'un texte des informations sous la forme d'un ensemble d'entités qui seront ensuite utilisées pour remplir des bases de

données, faire du résumé, faire de l'indexation ou de la segmentation, répondre à des questions.

L'idée de synthétiser l'information d'un document en une base de données remonte au début des années cinquante (Zellig Harris [HAR51]). Une des premières implémentations de cette idée a été faite des années plus tard pour des textes médicaux à l'université de New York par Naomi Sager [SAG87]. D'autres projets ont suivi concernant par exemple la transformation d'encyclopédies entières sous forme structurée. L'EI telle qu'elle est conçue aujourd'hui s'est largement développée autour des travaux des équipes qui participent à une série de conférences dédiées à ce domaine : les *Message Understanding Conferences* [MUC3, MUC4, MUC5, MUC6, MUC7]. Il s'agit de traiter une tâche d'extraction limitée : l'analyse d'un ensemble de textes pour remplir des formulaires correspondant à une demande d'information spécifique. Typiquement un formulaire sera rempli par événement rencontré dans le corpus. Bien que cette tâche soit limitée, elle reste complexe, l'information peut être répartie à différents endroits d'un document, exprimée de différentes façons. Les systèmes qui ont été développés sont largement dédiés à une application spécifique et demandent une adaptation humaine coûteuse pour être utilisés sur différentes instances d'une même tâche. Un exemple est donné en figure 22. Il s'agit d'une tâche qui a fait l'objet d'une des premières compétitions MUC ([MUC3]). Le corpus est constitué de textes traitant d'attentats terroristes, pour chaque événement le système doit déterminer le type de l'attaque (bombe, fusil, ...), la date, le lieu, la cible et les dommages causés.

*19 March – A bomb went off this morning near a power tower in San Salvador leaving a large part of the city without energy, but no causalities have been reported. According to unofficial sources the bomb – allegedly detonated by urban guerrilla commandos – blew up a power tower in the northwestern part of San Salvador at 0650 (1250 GMT).*

<b>INCIDENT TYPE</b>	Bombing
<b>DATE</b>	March 19
<b>LOCATION</b>	El Salvador : San Salvador (city)
<b>PERPETRATOR</b>	urban guerrilla commandos
<b>PHYSICAL TARGET</b>	power tower
<b>HUMAIN TARGET</b>	-
<b>EFFECT ON PHYSICAL TARGET</b>	destroyed

Figure 22. Un rapport sur un attentat terroriste et une fiche d'information extraite.

D'après la terminologie établie dans MUC, la spécification d'un événement particulier et les relations à extraire sont désignées par un *scénario*. Ainsi nous distinguons un *domaine* général, comme la finance, et un scénario particulier comme la vente d'une marchandise. Les informations extraites sont appelées un *patron*.

#### 4.2.1 Les techniques de bases

Nous allons nous placer dans la perspective des travaux développés dans les conférences MUC. Pour cette tâche d'extraction, un processus d'EI est composé de trois

parties distinctes. La première est l'étiquetage par des analyses lexicales et morphologiques, elle est suivie d'une extraction, de groupes de mots qui représentent des faits pertinents pour la demande d'information, (cf. figure 23). Ces groupes de mots sont intégrés pour former des entités plus larges qui correspondent à des faits plus complexes (e.g. ensembles de mots liés par une relation prédéfinie comme une personne et une position liées par l'action « licenciement ». Dans la dernière étape, ces entités servent à générer un formulaire.

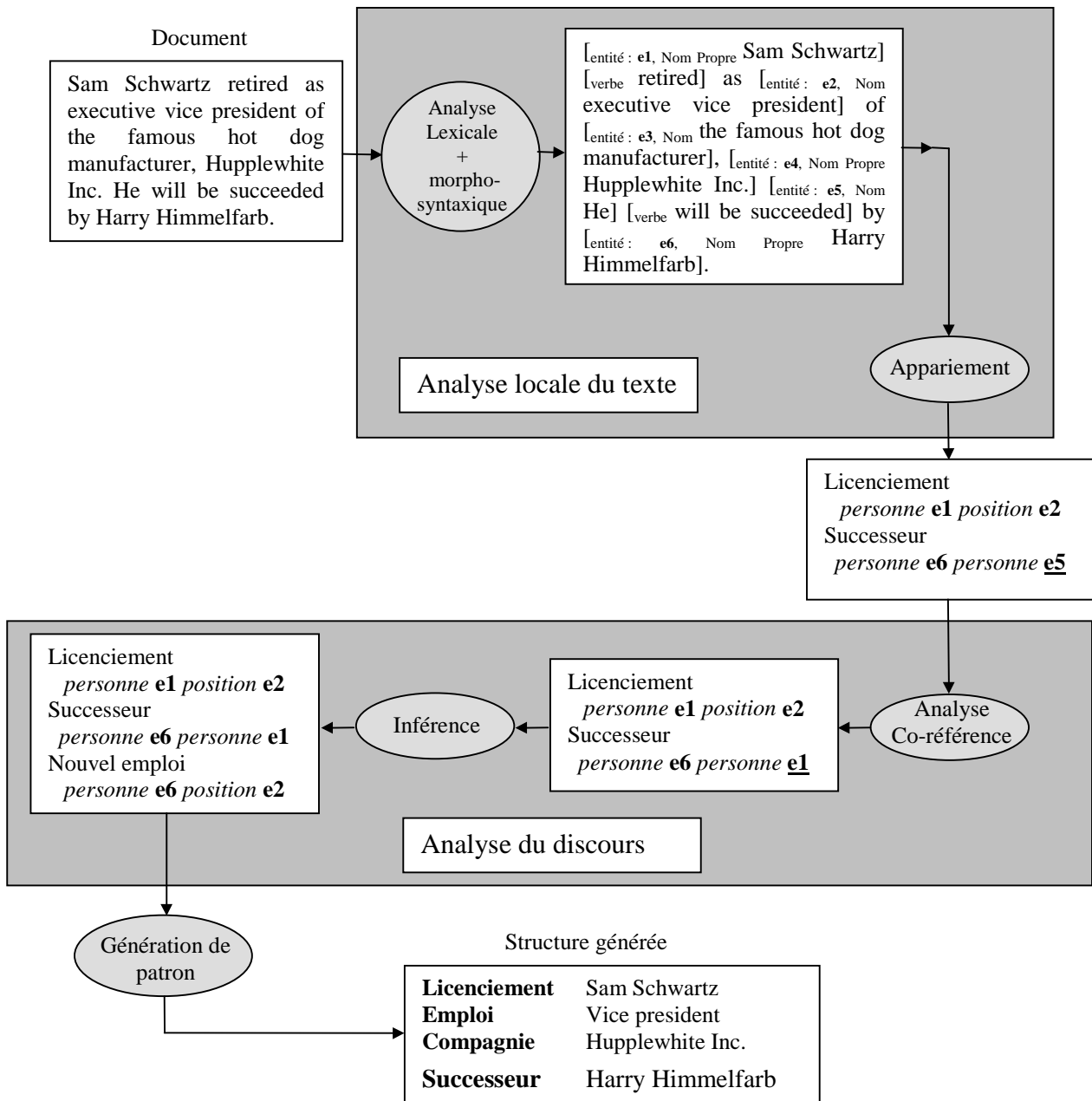


Figure 23. Structure d'un système d'Extraction de l'Information. L'exemple consiste à extraire à partir du document initial les informations concernant les mouvements de personnels sous la forme d'un formulaire qui est représenté en bas de la figure.



#### 4.2.1.1 Analyse locale du texte

Dans cette partie, le texte est d'abord divisé en phrases et en mots. On procède ensuite à une analyse lexicale puis à chaque mot est associée une étiquette morpho-syntaxique. Ceci peut se faire en utilisant, soit des étiqueteurs probabilistes [SCH94], soit des dictionnaires comme le *Complex Syntax dictionary* [GRI94].

La phase suivante identifie les entités nommées : différents types de noms propres ainsi que d'autres formes comme les dates ou les devises. Les noms sont importants pour beaucoup de tâches d'extraction. Les noms propres sont généralement identifiés par un ensemble d'expressions régulières comme les titres (M., Mme, Mlle), les suffixes (Jr.), ou autres. Les noms de sociétés sont identifiés par leur préfixe ou suffixe (Inc., Corporation, Associates). Pour ce faire, certains systèmes utilisent des dictionnaires de noms et d'autres apprennent des règles à partir d'un corpus annoté [BIK97].

Comme les arguments et les relations à extraire correspondent respectivement à des séquences de mots et à des relations fonctionnelles grammaticales, l'identification des structures syntaxiques du texte simplifie la problématique de l'extraction. Certains systèmes procèdent à une analyse de surface, *SRI FASTUS* [APP93, BAG97] par exemple utilise l'information syntaxique locale pour construire un étiquetage partiel des phrases, d'autres au contraire essaient de construire l'étiquetage et les relations grammaticales complètes d'une phrase.

A l'issue de ces analyses locales, des termes pertinents pour la tâche sont extraits en général par des automates déterministes construits de façon ad-hoc.

#### 4.2.1.2 Analyse du discours

Cette analyse est principalement constituée d'une analyse de co-référence puis d'une inférence sur les événements globaux extraits.

L'analyse de co-référence a pour but de résoudre les références anaphoriques. Dans l'exemple de la figure 23, le pronom « *he* » (entité e5) est remplacé par la plus récente entité du type *personne* mentionné avant lui (entité e1).

- La procédure d'inférence traite des problèmes *d'homogénéisation* : il y a dans le document des informations partielles sur un événement qui sont éparpillées tout au long du texte et qui ont besoin d'être combinés avant la génération du patron.
- Des problèmes *d'explicitation* : il existe des informations qui sont implicites et qui ont besoin d'être explicitées. Dans l'exemple de la figure 23, *Sam Schwartz* était président et il est remplacé par *Harry Himmelfarb*. C'est donc *Harry* qui est le nouveau président.

Ces inférences peuvent être implémentées par des systèmes à bases de règles, ou codées en dur dans le système [GRI92].

Les systèmes développés marchent bien si :

- A) L'information à extraire est exprimée d'une façon directe (il n'y a donc pas besoin d'une inférence compliquée),
- B) L'information est dans la majorité des cas exprimée par un assez petit nombre de formes,

- C) L'information est exprimée localement dans le texte (Bagga et Biermann, [BAG97] étudient le taux de performances des systèmes d'extraction en fonction de la localité de l'information, exprimée en termes de nombres de relations syntaxiques).

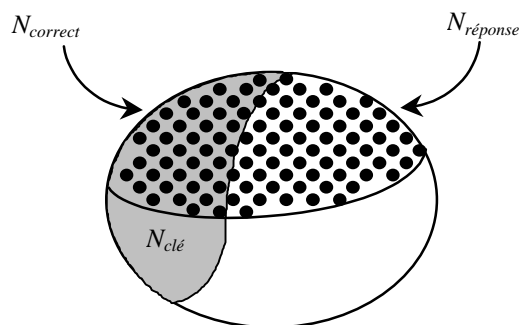
#### 4.2.2.2 Les mesures de Rappel/Précision et l'évaluation dans MUC

Dans la campagne d'évaluation MUC, une description détaillée d'un scénario (l'information à extraire) ainsi qu'un ensemble de documents et un patron à extraire pour chaque document (la base d'apprentissage) sont donnés aux participants.

Les équipes disposent alors d'un temps limité (de 1 à 6 mois) pour adapter leur système au nouveau scénario. Après ce temps, un nouvel ensemble de documents (la base test) est distribué aux participants afin qu'ils fournissent les patrons extraits par leur système pour ces documents.

Les systèmes sont alors évalués par une série de mesures dont les principales composantes sont le rappel et la précision.

Désignons par  $N_{clé}$  le nombre total des champs dans les patrons réponses établis par les experts,  $N_{réponse}$  le nombre total de champs remplis par le système, et  $N_{correct}$  le nombre de champs correctement remplis par le système, i.e. le nombre de champs qui coïncident avec les réponses des experts. Alors les mesures de précision et de rappel sont définies comme :



$$\text{Précision} = \frac{N_{correct}}{N_{réponse}}$$

$$\text{Rappel} = \frac{N_{correct}}{N_{clé}}$$

Il est aussi d'usage d'employer parfois la *mesure\_F*, définie comme :

$$\text{Mesure}_F = \frac{2 \times \text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

#### 4.2.2 Portabilité et Performance

##### *Portabilité*

Une des difficultés majeures pour faire de l'EI une technologie portable est le coût d'adaptation du système d'extraction à un nouveau scénario. En général, pour chaque nouveau scénario il faut des mois d'efforts pour développer un système. Les concepteurs de systèmes d'extraction se sont donc tournés vers de nouveaux outils leur permettant de créer et d'adapter rapidement un système. Comme la plupart de ces systèmes sont issus des compétitions MUCs, nous allons présenter leur évolution au cours de ces différentes compétitions.

Dans les premières MUC, les organisateurs mettaient à la disposition des participants de grosses bases d'apprentissage (plus de mille documents et leurs patrons associés pour MUC-2). Ces corpus ont encouragé le développement de systèmes dépendants du domaine traité.

Plusieurs équipes ont étudié la possibilité d'obtenir de l'information en examinant des phrases types. Dans *CIRCUS* par exemple (système développé pour MUC-3) Lehnert et al. [LEH91] ont étudié la possibilité de créer de l'information utile pour le remplissage de patrons en analysant un grand nombre de petites structures du type phrase. Pour MUC-4, ils ont développé *AutoSlog* [LEH92, RIL93] pour s'acquitter de cette tâche d'une manière semi-automatique en apprenant des règles d'extraction à partir des documents et de leur patron associé dans la base d'apprentissage.

La création de ce genre de corpus s'est avérée trop coûteuse, en effet les experts humains ne peuvent fournir qu'un nombre limité de patrons (de l'ordre d'une dizaine). On a alors assisté au développement de systèmes automatiques, utilisant des techniques d'apprentissage, pour traiter des bases d'apprentissage beaucoup plus réduites (cent documents pour MUC-7).

Par exemple, le système *CRYSTAL* créé pour MUC-6 [FIS95, SOD95] utilise des techniques d'apprentissage symbolique pour traiter ce problème sans intervention humaine. Mais la plupart de ces systèmes se sont révélés incapables de généraliser la connaissance de manière suffisante [FIS95]. Pour compenser le manque d'exemples en apprentissage les participants ont alors essayé d'apporter plus d'informations pour chaque document.

- Krupa, qui a développé le système *HASTEN* [KRU95] pour MUC-6, a construit par exemple une description structurelle pour chaque document, marquant le type de constituant, les contraintes sur les constituants ainsi que l'étiquette sémantique de chaque constituant. Cette approche permet d'obtenir de bonnes performances en généralisant la tâche MUC-6, mais demande en contrepartie quelques expertises sur les documents de la part du concepteur.
- Grishman, a élaboré un système d'extraction basé sur l'interaction utilisateur [GRI95]. L'utilisateur commence par donner un exemple ainsi que le patron à extraire. Le système construit à partir de la base une description détaillée de l'exemple. Il interagit alors avec l'utilisateur pour étendre et généraliser l'exemple. Il est apparu que dans les cas où l'utilisateur ne dispose pas d'une grande base annotée, ce genre de système pouvait apporter une approche très efficace pour l'acquisition et le traitement de l'information.

### *Performances*

Une autre difficulté qui a empêché l'utilisation grand public des systèmes d'extraction est la limitation de leurs performances. Les performances des systèmes dans les plus récentes compétitions n'excèdent pas 50% en rappel et 70% en précision [MUC7].

### **4.3. Recherche de l'information**

Le recherche d'information ou recherche documentaire est un domaine déjà relativement ancien (plus de 30 ans) qui s'est centré sur un ensemble de problèmes

d'accès à des documents pour répondre à des requêtes ouvertes ou fermées. Ces tâches de base sont encore d'actualité, par contre les corpus ont largement évolué depuis ces premiers travaux et de nouvelles tâches sont apparues. Comme nous l'avons déjà indiqué, le domaine connaît actuellement une évolution rapide. Il est déjà riche d'une longue histoire et de très nombreux modèles ou façons d'aborder les différentes tâches ont été proposées. Toutefois, le texte étant un objet complexe, souvent fortement bruité, ce sont très souvent des modèles assez simples et robustes qui sont à la base des systèmes les plus performants pour nombre de tâches de la RI. Il est fréquent que des systèmes plus sophistiqués, proposant des solutions plus satisfaisantes d'un point de vue théorique ou intuitif, prenant en compte des informations plus riches sur le texte que les modèles de base que l'on vient d'évoquer n'apportent que peu d'amélioration pour un coût nettement supérieur. Ce trait caractéristique du domaine mérite réflexion quand on se lance dans la conception de nouveaux modèles. Dans cette section, nous allons rappeler brièvement tout d'abord les tâches les plus classiques de la recherche d'information puis les grandes approches qui ont été développées pour apporter des solutions.

### **4.3.1 Tâches classiques en RI**

#### **4.3.1.1 Requêtes Ad-hoc, Routage et Filtrage**

Les tâches classiques de la RI consistent à apparier une requête à une collection de documents et à retourner les documents pertinents à l'utilisateur. Le succès du système dépend en partie de la qualité et de la quantité d'informations associées à la requête. En effet avec une grande quantité d'informations définissant la pertinence des documents, le système sera en mesure d'employer des techniques plus avancées pour classer les documents pertinents et non-pertinents. Nous introduisons ci-dessous trois tâches de base de la RI qui sont : la recherche *adhoc*, le routage et le filtrage [HUL94].

##### ***Adhoc***

Dans la recherche *adhoc*, l'utilisateur pose des requêtes auxquelles le système est censé répondre en lui renvoyant un ensemble de documents ordonnés selon une mesure de pertinence. On considère en général que le corpus est fixe et que les requêtes sont totalement ouvertes.

##### ***Routage***

Ici les requêtes sont fixes, et il s'agit d'ordonner un ensemble de documents par rapport à ces requêtes. Il s'agit d'un problème de classification pour lequel on désire ordonner les documents par rapport à chaque classe. Les corpus vont en général être dynamiques, i.e. ils peuvent être volatiles, évoluer, se modifier (e.g. messageries, forums électroniques, etc).

##### ***Filtrage***

Le but d'un système de filtrage est de trier parmi un large volume d'informations générées dynamiquement et de présenter à l'utilisateur celles qui se rapprochent le plus à sa requête. C'est aussi une tâche de classification, mais cette fois-ci on se contente de

prendre une décision sur la pertinence d'un document pour une classe. A la différence du routage, il s'agit d'une décision binaire.

#### 4.3.1.2 Classification de Texte

La classification de textes est une tâche importante en RI. Elle consiste à classer les documents dans un ensemble de classes prédéfinies. Une des premières méthodes de classification de textes est l'algorithme de Rocchio [ROC71] basé sur le modèle de recherche vectoriel (section 4.3.3.2). Cet algorithme génère, pour chaque classe, un classifieur binaire en sommant les vecteurs d'une classe de documents et en soustrayant à cette somme les vecteurs des autres classes de documents. Cette procédure génère un vecteur pondéré final qui peut être alors utilisé pour classer les nouveaux vecteurs document en calculant un produit scalaire et en utilisant un seuil approprié. Plus récemment de nombreuses méthodes ont été appliquées à la classification de texte, que ce soient des techniques générales de discrimination ou des adaptations de ces méthodes au cas du texte.

Parmi les techniques fréquemment employées, on retrouve les arbres de décision comme C4.5 [QUI93]. Wiener et al. [WIE95] ont étudié l'application des réseaux de neurones pour la détection thématique, qui peut être vue comme une forme de classification. Cette étude a été importante pour comprendre les avantages et les limitations de l'application de ce genre de méthodes au texte. Sur cette base, beaucoup de méthodes pour la catégorisation de textes ont été développées et un très grand nombre d'études comparent les différentes représentations et algorithmes. Schutze et al. [SCS95] font une excellente comparaison des différentes méthodes de classification incluant la régression logistique, l'analyse discriminante linéaire et les réseaux de neurones. Ils comparent aussi une représentation vectorielle simple avec le LSI [DEE90] (*Latent Semantic Indexing*) et ils observent qu'une combinaison redondante de ces deux représentations semble obtenir les meilleures performances au niveau de leurs expériences. Lewis et al. [LEW96] comparent différentes méthodes d'apprentissage pour la classification de texte. Ces méthodes incluent l'algorithme de Rocchio, la règle d'apprentissage de Widrow-Hoff et l'algorithme du gradient conjugué. Ils trouvent que ces deux dernières méthodes sont plus performantes que l'algorithme de Rocchio.

Plus récemment, les machines à vecteurs supports ont été employées pour la classification de texte. Joachims [JOA97] compare les MVS et plusieurs autres méthodes d'apprentissage pour la classification de texte, dont Naïve Bayes, les arbres de décision et les plus proches voisins. Il conclut que les MVS donnent de meilleurs résultats en terme de Précision et de Rappel. Malheureusement, il change les paramètres des systèmes de classification après examen de leurs performances sur la base test, ce qui soulève quelques questions quant à la validité de ses résultats. Dumais et al. [DUM98] comparent aussi les MVS à Rocchio, aux classifieurs bayésiens et aux arbres de décisions. Ils trouvent encore que les MVS donnent de bons résultats.

Une autre classe de techniques intéressantes est constituée par les réseaux Bayésiens. Sahami [SAH98] les utilise pour déterminer une hiérarchie entre les documents. Son réseau bayésien est une structure contenant un nœud  $C$  représentant la variable « classe de document » et un nœud  $X_i$  pour chaque caractéristique de la forme en entrée (i.e. les

termes dans le cas des documents). Pour un vecteur  $x$  en entrée, le réseau donne une estimation de la probabilité a posteriori  $p(C=c_x/X=x)$  pour chaque classe et détermine la classe de  $x$  comme celle qui maximise  $p(C/X=x)$ .

Le classifieur le plus simple et le plus étudié est probablement le classifieur Naïve Bayes (section 3.4.1.1, chapitre 3) qui peut être vu comme un cas particulier des réseaux Bayesiens. Ce classifieur fait l'hypothèse qu'étant donnée une classe, toutes les caractéristiques sont indépendantes les unes des autres et de ce fait :

$$p(X / C) = \prod_i p(X_i / C)$$

Cette hypothèse correspond au réseau Bayésien montré dans la figure 24.

Bien que les hypothèses statistiques d'indépendance ne soient jamais vérifiées en pratique, Lewis [LEW92] a montré que l'utilisation du classifieur *Naïve Bayes* est efficace dans le domaine du texte. Il montre en outre que ce genre de classifieur probabiliste simple est plus performant que les arbres de décision qui sont pourtant capables d'estimer les dépendances probabilistes entre les mots d'un document.

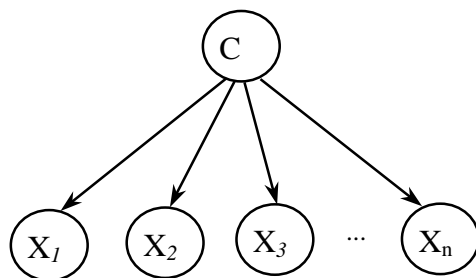


Figure 24. La structure d'un réseau Naïve Bayes

Il est à noter que le terme de « classifieur *Naïve Bayes* » a été beaucoup utilisé au sein de la communauté de classification de textes et englobe deux modèles probabilistes différents. Alors que ces deux modèles sont très similaires pour l'application de la décision Bayésienne et pour l'utilisation des hypothèses d'indépendance, ils diffèrent sur leur formulation probabiliste. Ces modèles sont étudiés dans McCallum et Nigam [MCN98]. Ils modélisent l'apparition des mots soit par une loi de Bernoulli, soit par une distribution multinomiale.

#### 4.3.2 Analyse automatique du texte et Architecture d'un système de RI

La majorité des systèmes de RI repose sur une approche statistique plutôt que sur des méthodes issues du traitement automatique du langage. Il y a plusieurs raisons que l'on peut avancer pour cela. Cet état de fait peut sembler contre intuitif car il semble évident que les connaissances sur le langage doivent jouer un rôle essentiel dans le développement des systèmes de recherche intelligents sur du texte. Spark Jones [SpJ73] discute de l'utilisation des méthodes linguistiques dans l'analyse automatique du texte ainsi que de l'espace minimal requis pour la représentation des connaissances syntaxiques et sémantiques. Elle conclut que non seulement ces méthodes sont trop dépendantes du domaine mais qu'en plus elles nécessitent beaucoup d'espace mémoire

pour le stockage de l'information. Une deuxième raison est que l'analyse linguistique est coûteuse à implémenter et demande plus de travail « humain ». Une dernière raison que l'on peut mettre en avant est que l'approche statistique qui a été développée depuis de nombreuses années, Luhn [LUH58] par exemple est un des pionniers du domaine, a montré son efficacité et sa simplicité de mise en œuvre. Nous allons présenter dans ce qui suit tout d'abord les principales étapes de l'indexation qui permet d'obtenir une description d'un document écrit en langage naturel puis un ensemble de modèles permettant de calculer les scores de documents, qui sont classiques en RI.

#### 4.3.2.1 Indexation

L'indexation de documents est obtenue en extrayant du texte un ensemble de termes représentatifs. Une procédure d'extraction basique de termes est composée de quatre étapes principales : d'abord on convertit les mots en minuscules sauf éventuellement certains mots comme les noms propres. Ensuite on extrait les mots (une suite de caractères). Puis, une liste *ante-dictionnaire* est utilisée pour filtrer les mots, et enlever en particulier les articles et les prépositions. Enfin les mots restants sont normalisés par exemple en enlevant les suffixes. La plupart des systèmes effectuent ces étapes, d'autres traitements plus sophistiqués peuvent être utilisés pour construire des index plus riches, par exemple en identifiant des suites (en général des couples ou triplets) de mots qui constituent des expressions.

##### *Ante-dictionnaire*

Les mots les plus fréquents dans un langage n'apportent pas en général une grande quantité d'information (*stop words en anglais*). Ainsi les 200 à 300 mots les plus fréquents ne sont pas retenus dans la procédure d'indexation, ils sont réunis dans un *ante-dictionnaire*. Il existe des *ante-dictionnaires* dépendants du domaine et d'autres qui en sont indépendants. Un exemple de *stop word* dépendant du domaine est le mot « projet » dans le cas où la collection de documents est constituée de descriptifs de projets. Des exemples d'*ante-dictionnaires* en anglais, indépendant du domaine, sont donnés par Van Rijsbergen [VaR79] ou par Salton dans SMART [SAL71].

##### *Normalisation de mots*

La normalisation de mots a pour but d'obtenir la forme *normale* des mots ayant la même racine. Ainsi « dynamique », « dynamiquement » et « dynamiques » sont transformées en une même forme normale « dynamiq ». Il existe deux grands types de procédés de normalisation: normalisation par application de règles linguistiques, obtenue par exemple en anglais grâce à l'algorithme de réduction de Porter [POR80], et normalisation basée sur un dictionnaire. L'utilité de la normalisation et sa mise en œuvre est bien sûr fortement dépendante de la langue.

##### *Description d'un document*

La plupart des systèmes de recherche ne tiennent pas compte de la structure interne du document. La description (du document ou de la requête) la plus souvent utilisée repose sur une liste de mots (*bag of words*) issue de l'étape d'indexation [SAL70]. A partir de

cette description on construit une représentation vectorielle qui sera soit un vecteur binaire dont chaque composante marque la présence ou l'absence d'un terme dans le document, soit un vecteur numérique dont chaque composante exprime la fréquence d'apparition d'un terme d'index dans le document.

#### 4.3.2.2 Architecture d'un système de RI

Classiquement, un système de RI est composé de deux grandes composantes.

- Un procédé d'indexation : les documents et les requêtes sont décrits comme des vecteurs du même espace vectoriel. Nous noterons  $\vec{d}$  et  $\vec{q}$  les descriptions vectorielles d'un document  $d$  et d'une requête  $q$ .
- Une mesure de similarité entre chaque élément de l'ensemble des documents  $\{d\}$  et la requête  $q$ . La méthode la plus classique consiste à calculer le cosinus de l'angle entre  $\{\vec{d}\}$  et  $\vec{q}$ .

Les documents sont ensuite triés dans l'ordre décroissant de leur mesure de similarité. L'utilisateur a alors la possibilité d'interagir avec le système (feedback) et peut modifier sa requête au cours de la recherche, quelquefois, ce retour peut être automatisé. Un exemple historique de système de recherche en-ligne est le système MEDLINE [McC73]. La figure 25, montre les étapes décrites plus haut.

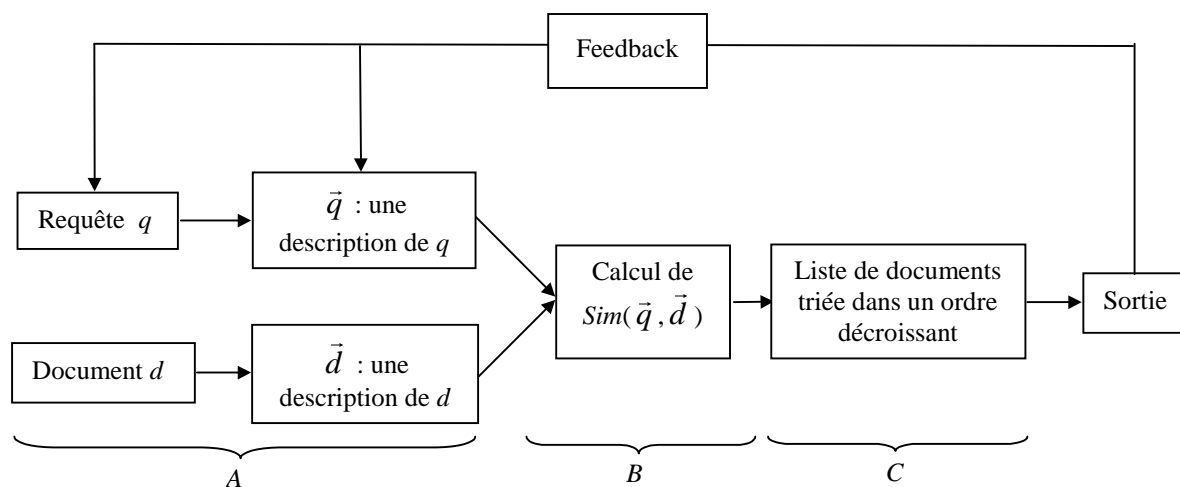


Figure 25. principales composantes d'un système de RI, A) Indexation, B) Appariement, C) Décision D) retour soit par interaction utilisateur soit de façon automatique

#### 4.3.3 Les Stratégies de Recherche en RI

Nous présentons maintenant les techniques de recherche classiques utilisées en RI basées sur les modèles booléens, les modèles vectoriels et les modèles probabilistes.



### 4.3.3.1 Modèles Booléens

Ces modèles utilisent une représentation booléenne des documents, qui caractérise simplement la présence ou l'absence des termes dans le document. Celui-ci sera décrit par un vecteur binaire sur l'espace des termes.

Dans la recherche booléenne, le système sélectionne les documents qui satisfont une expression logique sur les termes de la requête. Les opérations de bases pour ces modèles sont les connecteurs logiques : ET, OU et NON. La requête dans ce formalisme est exprimée sous la forme d'une expression logique comme  $Q = (A \text{ OU } B) \text{ ET } C$ , ce qui permet de trouver tous les documents contenant le terme  $C$  et un des termes  $A$  ou  $B$ . De nombreux moteurs de recherche ainsi que les catalogues de bibliothèques reposent sur cette stratégie de recherche.

La plupart des systèmes à base de règles utilisent essentiellement un modèle de recherche booléen [APT94, COH96]. Hull [HUL94] discute en détail les inconvénients de cette stratégie; les deux plus importants sont la difficulté pour modifier les requêtes complexes, et leur non-robustesse vis à vis du choix des termes de la requête.

Trouver la bonne combinaison des termes de la requête pour la recherche documentaire n'est pas simple. En particulier, plus la requête devient complexe, plus le nombre de combinaisons des connecteurs augmente rapidement. Dans ce formalisme tous les termes de la requête sont de même importance.

### 4.3.3.2 Modèles Vectoriels

Les modèles vectoriels regroupent un grand nombre de méthodes de recherche. La requête et les documents sont indexés en deux étapes. D'abord, les termes appropriés  $m_i$  sont extraits de la requête  $q$  ou du document  $d_j$ . Ensuite, on donne à chaque terme un poids qui reflète son importance :

$$\vec{d}_j = (a_{0,j}, \dots, a_{i,j}, \dots, a_{m-1,j})^T$$
$$\vec{q} = (b_0, \dots, b_i, \dots, b_{m-1})^T$$

$a_{i,j}$  et  $b_i$  désignent respectivement les poids du terme  $m_i$  dans le document  $d_j$  et la requête  $q$ . Si un terme indexé n'apparaît pas dans le document ou la requête, son poids est fixé à zéro.

Un score est généré par une fonction de similarité à partir de la représentation de la requête et du document. Pour chaque requête, les documents sont présentés à l'utilisateur dans l'ordre décroissant des scores. L'utilisateur inspecte les documents ordonnés et peut donner son avis sur leur pertinence. L'information sur la pertinence peut être utilisée par le système pour calculer une nouvelle description de la requête.

Dans ce qui suit, nous allons présenter un certain nombre de modèles vectoriels de recherche testés par Salton et Buckley [SAL88], qui sont deux pionniers du domaine. Posons  $\Phi = \{m_i\}_i$  l'ensemble des termes indexés pour une collection de documents et une requête donnée. La représentation la plus courante dans ces modèles est le codage *tf-idf*, nous en donnons ci-dessous une définition et en présentons deux variantes.

### **Fonction de similarité en Cosine avec la pondération *tf-idf***

Posons,  $tf(m_i, d_j)$  la fréquence du terme  $m_i$  dans le document  $d_j$ ,  $tf(m_i, q)$  la fréquence du terme  $m_i$  dans la requête  $q$  et  $idf(m_i)$  l'inverse du nombre de documents dans la collection contenant le terme  $m_i$ . Dans le codage *tf-idf*, les vecteurs représentatifs du document  $d_j$  et de la requête  $q$  s'écrivent respectivement :

$$\begin{aligned}\vec{d}_j &= (a_{ij})_{i=\{0,\dots,m-1\}} \\ \vec{q} &= (b_i)_{i=\{0,\dots,m-1\}}\end{aligned}$$

avec  $a_{ij}=tf(m_i,d_j).idf(m_i)$  et  $b_i=tf(m_i,q).idf(m_i)$ . Si aucune information supplémentaire n'est disponible sur la collection de documents, ni sur la variabilité de la taille des documents, la mesure de similarité la plus adaptée pour la RI est la mesure de cosinus définie par :

$$Sim(\vec{q}, \vec{d}_j) = \frac{\vec{d}_j \cdot \vec{q}}{\|\vec{d}_j\| \times \|\vec{q}\|}$$

De très nombreuses variantes du codage *tf-idf* et des fonctions de similarités ont été proposées, elles ont fait l'objet d'un grand nombre de comparaisons expérimentales. Nous présentons deux variations autour de *tf-idf*.

#### **Pondération logarithmique des termes**

Dans le cas de requêtes et de documents de grandes tailles par exemple, la pondération *tf-idf* linéaire devient trop forte lorsque les caractéristiques fréquentielles des termes sont grandes. Buckley et al. [BUC92], ont introduit une pondération logarithmique fréquentielle pour diminuer la disparité entre ces caractéristiques.

La mesure de similarité utilisée dans ce cas sera la même fonction cosinus que précédemment et les composantes des vecteurs  $d_j$  et  $q$  deviennent :

$$\begin{aligned}a_{ij} &= (1 + \log(tf(m_i, d_j))) \\ b_i &= (1 + \log(tf(m_i, q))) \cdot idf(m_i)\end{aligned}$$

Cette pondération non linéaire a été justifiée par Robertson [ROB94] comme la moyenne de 2 modèles de Poisson.

#### **Pondération logarithmique normalisée**

Pour tenir compte des documents de petites tailles, Singhal et al. [SIN96] ont introduit une nouvelle mesure de normalisation. Ils normalisent la caractéristique logarithmique fréquentielle  $1 + \log(tf(m_i, d_j))$  par une quantité qui dépend de la moyenne des caractéristiques fréquentielles :  $moy\_tf(d_j) = \sum_i tf(m_i, d_j) / |d_j|$ , et ils définissent le facteur de normalisation comme l'interpolation entre la taille du document  $l_j = |d_j|$  et la

taille moyenne des documents  $\Delta = \sum_j l_j$ . Dans leur étude, la fonction de similarité est un simple produit scalaire,  $Sim(\vec{q}, \vec{d}_j) = \vec{d}_j \cdot \vec{q}$ . Où les composantes des vecteurs sont :

$$a_{ij} = \frac{1 + \log(tf(m_i, d_j))}{1 + \log(moy\_tf(d_j))}$$

$$b_i = (1 + \log(tf(m_i, d_j))) \cdot idf(m_i)$$

Les compétitions TREC ont permis de caractériser le potentiel de ces différents codages et mesures sur des grands corpus, certaines versions employées dans des grands systèmes de RI sont devenues des références en la matière.

#### 4.3.3.3 Modèles Probabilistes

Parallèlement à cette approche « métrique », plusieurs auteurs ont développé des approches probabilistes pour la recherche documentaire. Ces méthodes essaient de capturer la distribution des mots dans les documents et s'en servent pour faire de l'inférence.

La première étude sur ces modèles remonte au début des années soixante [MARo60] et depuis, de nombreux autres modèles ont été proposés. Le score utilisé dans ces modèles est la probabilité de pertinence par rapport à une requête. Une des difficultés avec cette approche est d'estimer des probabilités ou des densités dans les espaces de grande dimension correspondant à l'espace des termes en RI. Tous les modèles font des hypothèses drastiques pour permettre des estimations robustes et tout simplement rendre possible les calculs. Les premiers modèles proposés supposent une indépendance complète des termes du document. La nécessité de ces hypothèses pour mener à bien les calculs a donné lieu à de très nombreux travaux qui ont conduit à proposer des modèles plus sophistiqués qui font des hypothèses moins fortes ou qui emploient d'autres types de méthodes. Si on note  $P_q$  une variable aléatoire binaire qui indique la pertinence d'un document pour une requête, alors les modèles probabilistes utiliseront comme score la probabilité de pertinence d'un document pour une requête  $p(P_q / \vec{d}, \vec{q})$  ou d'autres quantités liées à celle-ci.

Une des justifications avancées pour l'usage des modèles probabilistes est le *#probability ranking principle#* [ROB76]. Enoncé de façon informelle, ce principe dit que « les performances optimales pour la recherche seront obtenues quand les documents sont fournis de façon ordonnée selon leur probabilité de pertinence pour une requête ». Dans le contexte de l'approche probabiliste, les notions de « pertinence », « optimalité » et « performances » peuvent être définies de façon exacte. Nous ne développerons pas plus ce point ici.

Des travaux fondateurs pour les modèles probabilistes sont ceux de Robertson et Sparck-Jones [ROB76] et de Croft et Harper [CRO79] qui ont développé les idées de Marron et Kuhns [MARo60]

[ROB76] par exemple travaillent sur des documents représentés de façon binaire. Pour une requête donnée  $q$ , ils montrent que les documents pertinents sont ceux qui vérifient :

$$\frac{p(\bar{d} / P_q, \bar{q})}{p(\bar{d} / \overline{P_q}, \bar{q})} > \frac{\lambda_2 p(\overline{P_q})}{\lambda_1 p(P_q)}$$

où  $\overline{P_q}$  est l'événement « non-pertinent » et  $\lambda_1, \lambda_2$  sont respectivement les coûts de ne pas déclarer pertinent un document pertinent et réciproquement.

Les densités  $p(d/z)$  sont estimées en faisant une hypothèse d'indépendance des termes  $a_j$  du document :  $p(d / z) = \prod_j p(a_j / z)$ .<sup>11</sup>

De nombreuses variantes de ce modèle ont été proposées par la suite et reposent sur des idées légèrement différentes sur les rôles respectifs des requêtes et documents. Un autre travail essentiel dans cette famille d'approches est celui de [FUH89].

Les modèles de mélange de densité ont été également largement explorés. L'idée est que la distribution d'un terme dans un document va être fonction du fait que le document soit pertinent ou pas. Les distributions des termes sont modélisées par des mélanges. Le modèle le plus connu est le modèle 2-Poisson [BOO74].

Une autre approche probabiliste a été popularisée par le réseau d'inférence INQUERY proposé par Turtle et Croft [TUR91]. Comme dans [FUH89], [TUR91] intègrent les modèles d'indexation et de recherche en faisant des inférences sur les concepts à partir des caractéristiques. Les caractéristiques incluent des mots, des groupes de mots et des caractéristiques structurées plus complexes. En utilisant un réseau bayésien, ils infèrent la probabilité que la requête soit satisfaite par un document.

Ponte et Croft [PON98] proposent une méthode non paramétrique intégrant l'indexation et la recherche documentaire en un seul modèle. Ils construisent un modèle par document et estiment la probabilité que la requête ait pu être générée par chacun de ces modèles. Les documents sont ensuite ordonnés par rapport à leur probabilité.

Plus récemment, des techniques basées sur les modèles de Markov cachés ont été introduites pour différentes tâches de RI. Miller et al. [MIL99] par exemple proposent un tel modèle pour la recherche ad-hoc. L'idée de base de cette modélisation est d'utiliser un MMC ergodique à deux états dont le premier représente le fait qu'un mot de la requête soit généré par le document et le second, le fait que ce mot soit généré par un ensemble plus général de termes.

---

<sup>11</sup> L'hypothèse faite en réalité dans ce modèle est un peu moins forte [CRE98].

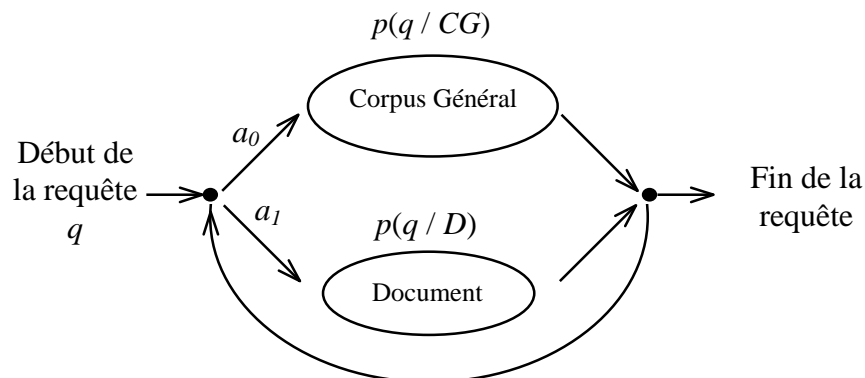


Figure 26. MMC ergodique à deux états pour la recherche documentaire

Les probabilités de transitions et d'émission sont calculées par l'algorithme *EM* en utilisant le document et la collection. La figure 26, résume l'architecture du MMC utilisé. Sur ce modèle, des hypothèses contraignent certaines des probabilités de transition à être égale, ce qui simplifie considérablement les calculs. Par exemple, la formule pour le calcul de  $p(q / D \text{ est pertinent})$ , correspondante à la figure 26 est simplement :

$$p(q / D \text{ est pertinent}) = \prod_{q_i \in q} (a_0 p(q_i / CG) + a_1 p(q_i / D))$$

Ce modèle permet de prendre en compte de façon limitée l'ordre séquentiel des mots dans un document. Des versions un peu plus sophistiquées de ce modèle ont obtenu d'excellents résultats sur la tâche ad-hoc de TREC 6.

Dans cette description, nous n'avons fait qu'évoquer quelques-uns des modèles probabilistes pour la recherche d'information. Ce domaine a été très largement exploré, et il n'est d'ailleurs pas toujours très simple de s'y retrouver dans cet ensemble de méthodes. Une discussion synthétique détaillée des modèles probabilistes peut être trouvée dans [CRE98].

Les modèles vectoriels et probabilistes ont chacun leurs partisans. Dans la pratique, des systèmes reposant sur l'une ou l'autre des approches ont des performances similaires. Les modèles probabilistes permettent quand même de mieux expliciter les hypothèses utilisées et de donner un sens aux règles de décision.

#### 4.3.4 Techniques de Recherche Interactives et retour utilisateur

Les techniques interactives de recherche jouent un rôle important pour les systèmes de RI. Elles reposent principalement sur des jugements utilisateurs portant sur les résultats de la recherche, sur des techniques de visualisation de corpus permettant la navigation. Nous allons traiter ici du *retour utilisateur (feedback)* qui est une technique classique par laquelle un utilisateur et le système interagissent afin d'améliorer la qualité de la

recherche [SAL90]. Cette interaction permet classiquement de modifier la requête puis de la re-soumettre au système.

Les techniques de *feedback* reposent sur la notion de pertinence. Dans le processus de communication, la pertinence est considérée comme une mesure de la correspondance entre une source et une destination [SAR75]. Nous la considérons comme une variable aléatoire qui prend la valeur 1 (pertinent) ou la valeur 0 (non-pertinent). Nous allons présenter deux méthodes d'utilisation du *feedback*; les deux techniques obéissent au même formalisme mathématique.

#### 4.3.4.1 Formalisme mathématique du feedback

Posons :

1.  $j : \{1, \dots, n\} \rightarrow \{0, \dots, n-1\}$ ,  $r \mapsto j(r)$  une fonction bijective telle que  $Sim(\vec{q}, \vec{d}_{j(1)}) \geq Sim(\vec{q}, \vec{d}_{j(2)}) \geq \dots \geq Sim(\vec{q}, \vec{d}_{j(n)})$ .
2.  $D_r(q) = \{d_{j(1)}, \dots, d_{j(r)}\}$ , l'ensemble des  $r$  documents ayant la similarité la plus grande, retourné par le système et inspectés par l'utilisateur.
3.  $D_r^{per}(q)$  et  $D_r^{non}(q)$  une partition de  $D_r(q)$ .  $D_r^{per}(q)$  contient les documents qui sont pertinents pour  $q$ , et  $D_r^{non}(q)$  les documents qui sont non-pertinents pour  $q$ .

Lorsque l'on considère les ensembles  $D_r^{per}(q)$  et  $D_r^{non}(q)$  pour une requête donnée, nous pouvons omettre la variable  $q$  et écrire simplement  $D_r$ ,  $D_r^{per}$  et  $D_r^{non}$ .

A partir de la description  $\vec{q}$  de la requête  $q$  et des ensembles  $D_r^{per}(q)$  et  $D_r^{non}(q)$ , on peut construire une nouvelle description  $\vec{q}'$  de la requête, i.e.

$$\vec{q}' = f\left(\vec{q}, \left\{ \vec{d}_j / d_j \in D_r^{per} \right\}, \left\{ \vec{d}_k / d_k \in D_r^{non} \right\} \right)$$

#### 4.3.4.2 Ré-estimation des poids des composantes de la requête

Robertson et Sparck-Jones, [ROB76] ont proposé de changer le poids d'un mot  $m_i$  appartenant à l'ancienne requête  $q$ , en  $\log\left(\frac{p_i(1-q_i)}{q_i(1-p_i)}\right)$  où  $p_i$  est l'estimation de probabilité qu'un document pertinent contienne  $m_i$ , i.e.

$$p_i = \frac{\# \text{ documents } d_j \in D_r^{per} \text{ contenant } m_i}{\# \text{ documents } d_j \in D_r^{per}}$$

et  $q_i$  est l'estimation de probabilité qu'un document non pertinent contienne  $m_i$ , i.e.

$$q_i = \frac{\# \text{ documents } d_j \in D_r^{non} \text{ contenant } m_i}{\# \text{ documents } d_j \in D_r^{non}}$$

#### 4.3.4.3 Expansion automatique de la requête et réestimation des poids de la requête

Une autre approche classique donnée par Rocchio [ROC71] propose de déplacer la description vectorielle de la requête vers la description vectorielle des documents pertinents tout en l'éloignant de la description vectorielle des documents non-pertinents. D'après le formalisme de Rocchio et en utilisant le formalisme ci-dessus, la nouvelle description de la requête s'écrit :

$$\vec{q}' = \alpha \frac{\vec{q}}{\|\vec{q}\|} + \frac{\beta}{|D_r^{per}|} \sum_{\vec{d}_j \in D_r^{per}} \frac{\vec{d}_j}{\|\vec{d}_j\|} - \frac{\gamma}{|D_r^{non}|} \sum_{\vec{d}_k \in D_r^{non}} \frac{\vec{d}_k}{\|\vec{d}_k\|}$$

Les descriptions sont normalisées par leur normes, i.e. par  $\|\vec{q}\| = \sqrt{\sum_{i=0}^{m-1} b_i^2}$  et

$$\|\vec{d}_j\| = \sqrt{\sum_{i=0}^{m-1} a_{i,j}^2}.$$

#### 4.3.5 Evaluation

L'évaluation est bien sûr un problème crucial en RI sur lequel la communauté RI a consacré énormément de ressources. Les premières expériences sur le sujet remontent à la fin des années soixante avec les expériences de Cranfield [CLE66, SpJ81] et de SMART [SAL71]. Plus récemment Donna Harman [HAM95] dans le cadre du NIST anime depuis plusieurs années des compétitions pour évaluer les systèmes de RI sur de grandes collections de documents. La qualité d'un système de RI est très difficile à évaluer, il faudrait pour cela être capable de prendre en compte de nombreux facteurs subjectifs tels que le contexte de la recherche, les jugements humains etc .

La RI a développé des mesures quantifiables de l'efficacité d'un système, qui ne reflètent qu'en partie ces différents critères mais sont utilisés faute de mieux.

Une mesure qui est largement utilisée pour quantifier l'efficacité de la recherche est le graphe de la précision et du rappel [FRE91] que nous utiliserons également dans nos expériences. On trouvera d'autres mesures dans Hull [HUL94] et Harman [HAR95].

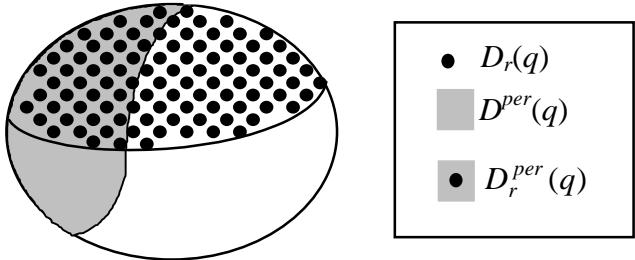
##### 4.3.5.1 Le Graphe de Précision et de Rappel

Une collection de test est constituée d'un ensemble de documents  $D$ , d'un ensemble de requêtes  $Q_0$  et des évaluations sur la pertinence des documents pour chaque requête. Les évaluations sur la pertinence divisent, pour chaque requête  $q$ , la collection  $D$  en deux sous-ensembles : les documents pertinents  $D^{per}(q)$  et les documents non-pertinents  $D^{non}(q)$ .

Supposons que les documents  $d_j \in D$ , soient triés dans l'ordre décroissant de leur mesure de similarité avec une requête  $q \in Q_0$ . Le graphe de précision et de rappel mesure comment les documents pertinents et non-pertinents sont répartis parmi les documents triés. L'utilisateur est supposé inspecter les  $r$  premiers documents de la liste triée. Cet

ensemble est appelé l'ensemble réponse  $D_r(q)=\{d_{j(1)}, \dots, d_{j(r)}\}$ . On calcule alors pour chaque ensemble réponse  $D_r(q)$  une valeur de précision et de rappel.

$$\text{Rappel}_r(q) = \frac{|D_r^{per}(q)|}{|D^{per}(q)|}$$

$$\text{Précision}_r(q) = \frac{|D_r^{per}(q)|}{|D_r(q)|}$$


Où  $D^{per}(q)$  est l'ensemble de tous les documents pertinents pour la requête  $q$ , et  $D_r^{per}(q) = D^{per}(q) \cap D_r(q)$  est l'ensemble de documents pertinents retournés par le système parmi les  $r$  premiers.

Le rappel mesure quelle est la proportion de documents pertinents retournés par le système et la précision le pourcentage de documents pertinents parmi ceux retournés.

Le tableau 1 montre l'évolution de ces mesures pour un exemple factice.

Une courbe linéaire par morceau est obtenue en reportant les points précision-rappel pour différentes valeurs de  $r$  (figure 27).

$r$	<i>Pertinent pour <math>q</math></i>	<i>Rappel(<math>q</math>)</i>	<i>Précision(<math>q</math>)</i>
1	+	$\frac{1}{4}=0.25$	$\frac{1}{1}=1.0$
2	-	$\frac{1}{4}=0.25$	$\frac{1}{2}=0.5$
3	-	$\frac{1}{4}=0.25$	$\frac{1}{3}=0.33$
4	+	$\frac{2}{4}=0.5$	$\frac{2}{4}=0.5$
5	-	$\frac{2}{4}=0.5$	$\frac{2}{5}=0.4$
6	-	$\frac{2}{4}=0.5$	$\frac{1}{3}=0.33$
7	+	$\frac{3}{4}=0.75$	$\frac{4}{7}=0.43$
8	+	$\frac{4}{4}=1.0$	$\frac{4}{8}=0.5$

Tableau 1. Un exemple jouet pour le calcul de la précision et du Rappel pour une requête  $q$  donnée et pour un nombre croissant de réponses.

Cette courbe est ensuite lissée en deux étapes. Dans la première, pour chaque requête  $q$ , on estime une fonction qui assigne à chaque valeur du rappel  $\rho \in [0,1]$  une valeur « plafond »  $\Pi_q(\rho)$  de précision telle que :

$$\Pi_q(\rho) = \max\{\text{Précision}_r(q) / \text{Rappel}_r(q) \geq \rho\}$$

La fonction  $\Pi_q(\rho)$  peut être interprétée comme le pourcentage de documents pertinents qu'un utilisateur observera s'il veut atteindre un taux de rappel d'au moins  $\rho$ .

Dans la deuxième étape, on calcule la fonction  $\Pi : [0,1] \rightarrow [0,1]$  obtenue en calculant la moyenne arithmétique de la fonction  $\Pi_q(\rho)$  pour l'ensemble des requêtes  $q \in Q_0$ .

$$\Pi(\rho) = \frac{1}{|Q_0|} \sum_{q \in Q_0} \Pi_q(\rho)$$



Le graphe de  $\Pi(\rho)$  pour  $\rho \in [0,1]$  est appelé le graphe de précision et de rappel.

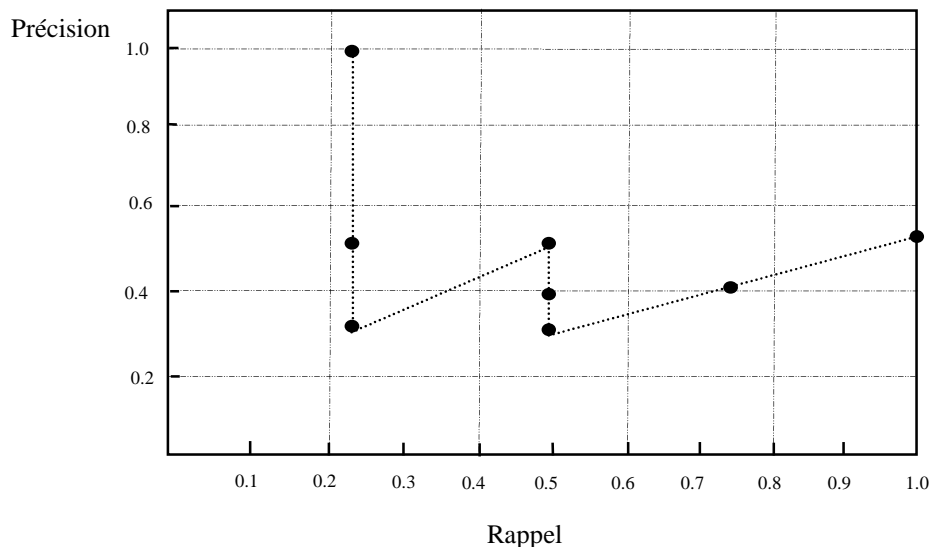


Figure 27. La courbe de rappel et de précision tracée à partir de l'exemple du tableau 1.

En pratique, les courbes de précision et rappel sont obtenues en calculant  $\Pi(\rho)$  pour les 11 valeurs de  $\rho = 0, 0.1, 0.2, \dots, 1.0$ . Le point de cette courbe correspondant aux valeurs égales de précision et de rappel est appelé le « *average point* ».

#### 4.3.5.2 Autres mesures de performance

D'autres mesures de performance basées sur le rappel et la précision sont possibles. Quelques études sur l'évaluation proposent de prendre le point de la courbe où la précision et le rappel sont égaux. Van Rijsbergen [VaR79] propose une mesure qui permet de pondérer l'importance relative de la précision et du rappel. Cette mesure, *F-stat*, se calcule de la façon suivante:

$$F - stat = \frac{Pr\acute{e}cision * Rappel}{\alpha * Rappel + (1 - \alpha) * Pr\acute{e}cision}$$

où  $\alpha$  varie entre 0 et 1 et mesure l'importance relative de la précision par rapport au rappel. D'autres mesures d'évaluation qui ne sont pas directement liées à la précision-rappel ont aussi été proposées. Mais il existe une grande corrélation entre la plupart de ces mesures.

## 4.4 Conclusion

Nous avons présenté dans ce chapitre des techniques de base utilisées en Extraction d'Information et en Recherche d'Information. Pour chaque domaine, nous avons décrit les tâches de base ainsi que les méthodes d'évaluation. Plusieurs des notions introduites

seront utilisées dans les chapitres 6 et 7. La description générale des problématiques EI et RI permet de situer notre travail dans le cadre de l'accès à l'information.

Dans le chapitre suivant, nous allons nous intéresser à la tâche de l'extraction de passages. Nous allons présenter plus particulièrement la tâche du résumé de texte, pour laquelle nous avons développé le système S.A.R.A.

## 4.5 Bibliographie

- [APP93] D. Applet, J. Hobbs, J. Bear, D. Israel, M. Tyson. FASTUS : A finite-state processor for Information Extraction from Real-World text. *In proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pp. 1172--1178, Août 1993.
- [APT94] C. Apte, F. Damerou, S. M. Weiss. Automated Learning of Decision Rules for Text Categorization. *Transactions of Office Information Systems*. Vol. 12, N° 3, 1994.
- [BAG97] A. Bagga, A. Biermann. Analyzing the Performance of Message Understanding systems. *Technical Report CS-1997-01*, Dept. of Computer Science, Université de Duke 1997.
- [BAH64] Y. Bar-Hillel. Language and Information. Selected Essays on their Theory and Application. *Addison-Wesley*, 1964.
- [BEL92] N. J. Belkin, W. B. Croft. Information Filtering and Information Retrieval: two sides of the same coin? *Communications of the ACM*, pp. 29--38, Vol. 35, N° 12, 1992.
- [BIK97] D. Bikel, S. Miller, R. Schwartz R. Weischedel. Nymble : a high-performance learning name-finder. *In Proceedings of the Fifth Applied Natural Language Processing Conference*, Washington DC, Avril 1997.
- [BOO74] A. Bookstein, D. Swanson. Probabilistic models for automatic indexing. *Journal of the American Society for Information Science*. Vol. 25, N° 5, pp.312--318, 1974.
- [BUC92] C. Buckley, G. Salton, J. Allan. Automatic Retrieval with Locality Information using SMART. *In TREC-1 Proceedings*, pp. 69--72, 1992.
- [CLE66] C. W. Cleverdon, E. M. Keen. Factors Determining the Performance of Indexing Systems. Vol2: Test Results. *Cranfield*, Angleterre: Aslib Cranfield Research Project.
- [COH96] W. W. Cohen. Learning trees and rules with set-valued features. *In Proceedings of the Thirteenth National Conference on Artificial Intelligence. AAAI96*, 1996.
- [CRA91] S. L. Crawford, R. M. Fung, L. A. Appelbaum, R. M. Tong. Classification Trees for Information Retrieval. *In Machine Learning: Proceedings of the Eighth International Workshop*. pp. 245--249, Morgan Kaufmann.
- [CRD98] M. Crave, D. Dipasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, S. Slattery. Learning to Extract Symbolic Knowledge from the World Wide Web. *In*

*Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI, 1998.*

- [CRE98] F. Crestani, M. Lalmas, C.J. Van Rijsbergen, I. Campbell. Is this document relevant? ... probably. *ACM Computing Surveys*, Vol. 30, N° 4, pp. 528--552, 1998.
- [CRO79] W.B. Croft, D.J. Harper. Using Probabilistic Models of Document Retrieval without Relevance Information. *Journal of Documentation*, pp. 285--295, vol. 35, 1979.
- [DEE90] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, pp. 391--407, Vol. 41, N° 6, 1990.
- [DUM98] S. T. Dumais. Using SVMs for text categorization. *IEEE Intelligent Systems Magazine*. Vol. 13, N° 4, 1998.
- [FIS95] D. Fisher, S. Soderland, J. McCarthy, F. Feng, W. Lehert. Description of the Umass system as used for MUC-6. *In Proceedings of Sixth Message Understanding Conference*. Morgan Kaufmann, 1995.
- [FRE91] H.P. Frei, S. Meienberg, P. Schäuble. The Perils of Interpreting Recall and Precision Values. *Informatik Fachberichte*, pp. 1--10, Vol. 289, Springer-Verlag, Berlin, 1991.
- [FUH89] N. Fuhr. Models for Retrieval with Probabilistic Indexing. *Information Processing and Management*. Vol. 25, N° 1, 1989.
- [GRI92] R. Grishman, C. Macleod, J. Sterling. New York University : Description of the Proteus system as used for MUC-4. *In Proceedings of the Fourth Message Understanding Conference*, pp. 233--241, McLean, Juin 1992.
- [GRI94] R. Grishman, C. Macleod, A. Meyers. Complex Syntax : Building a Computational Lexicon. *In proceedings 15<sup>th</sup> International Conference on Computational Linguistics (COLING'94)*. pp. 268--272, Kyoto, Japon, Août 1994.
- [GRI95] R. Grishman. The NYU system for MUC-6 or where's the syntax ? *In Proceedings of Sixth Message Understanding Conference*. Morgan Kaufmann, 1995.
- [HAM95] Evaluation Techniques and Measures. *In TREC-3 Proceedings*, pp. A-5-A-13, 1995.
- [HAR51] Zellig Harris. Structural Linguistics. *The University of Chicago Press*. 1951.
- [HAT75] S.P. Harter. A Probabilistic Approach to Automatic Keyword Indexing. *Journal of the American Society for Information Science*. 1975.

- [HUL94] D. Hull. Information Retrieval using Statistical Classification. *Thèse de doctorat*, Université de Stanford, 1994.
- [JOA97] T. Joachims. Text Categorization with Support Vector Machines: Learning with many Relevant Features. *Technical Report, LS8-Report, Universitaet Dortmund*, 1997.
- [KRU95] G. Krupa. SRA : Description of the SRA system as used for MUC-6. *In Proceedings of Sixth Message Understanding Conference*. Morgan Kaufmann, 1995.
- [LEH91] W. Lehnert, C. Cardie, D. Fisher, E. Riloff, R. Williams. University of Massachusetts : Description of the CIRCUS system as used for MUC-3. *In Proceedings of the Third Message Understanding Conference*. Morgan Kaufmann, 1991.
- [LEH92] W. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, S. Soderland. University of Massachusetts : MUC-4 test results and analysis. *In Proceedings of Fourth Message Understanding Conference*. Morgan Kaufmann, 1992.
- [LEW92] D. D. Lewis, Representation and Learning in Information Retrieval. *thèse de doctorat*, Université de Massachusetts, Department of Computer Science, 1992.
- [LEW96] D. D. Lewis, R. E. Schapire, J. P. Callan, R. Papka. Training Algorithms for Linear Text Classifiers. *In Proceedings of ACM-ASIGIR*, 1996.
- [LUH58] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, N° 2, pp. 159--165, 1958.
- [MAR60] M.E. Maron, K.L. Kuhns. On relevance probabilistic indexing and information retrieval. *Journal of the Associations of Computing Machinery*, N° 7, pp. 216--244, 1960.
- [MCN98] A. McCallum, K. Nigam. A comparison of event Models for Naive Bayes Text Classification. *In Learning for Text Categorization: Papers from the 1998 AAAI Workshop (Technical Report WS-98-05)*, M. Sahami Ed. 1998.
- [MCR98] A. McCallum, R. Rosenfeld, T. Mitchell, A. Ng. Improving text classification by shrinkage in a hierarchy of classes. *In ICML-98*, pp. 359--367, 1998.
- [McC73] D. B. McCarn, J. Leiter. On-line services in medicine and beyond. *Science*, N° 181, pp. 318--324, 1973.
- [MIL99] D.T.H. Miller, T. Leek, R.M. Schwartz. BBN at TREC7: Using Hidden Markov Models for Information Retrieval. *In Proceedings of the seventh Text Retrieval Conference, TREC-7*. NIST Special Publications, 1999.
- [MUC3] *Proceedings of the Third Message Understanding Conference*. Morgan Kaufmann, Mai 1991.

- [MUC4] *Proceedings of the Fourth Message Understanding Conference*. Morgan Kaufmann, Juin 1992.
- [MUC5] *Proceedings of the Fifth Message Understanding Conference*. Morgan Kaufmann, , Baltimore, Août 1993.
- [MUC6] *Proceedings of the Sixth Message Understanding Conference*. Morgan Kaufmann, Columbia, Novembre 1995.
- [MUC7] *Proceedings of the 7th Message Understanding Conference*. Morgan Kaufmann, 1998.
- [NIG00] K. Nigam, A. K. McCallum, S. Thurn, T. Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, pp. 103--134, Vol. 39, 2000.
- [PON98] J. M. Ponte, W.B. Croft. A Language Modeling Approach to Information Retrieval. *In Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval*. pp. 275--281, 1998.
- [POR80] M. F. Porter. An Algorithm for Suffixe Stripping. *Program*, Vol. 13, N° 3, pp. 130--137, 1980.
- [QUI93] J. R. Quinlan. C4.5: Programs for Machine Learning. *Morgan Kaufmann, Inc.* 1993.
- [RIL93] E. Riloff. Automatically constructing a Dictionary for information extraction tasks. *In Proceedings of 11<sup>th</sup> Annual Conference of Artificial Intelligence*, pp. 811-816, 1993.
- [ROB76] S. E. Robertson, K. Sparck-Jones. Relevance Weighting of Serach Terms. *Journal of American Society for Information Science*, N° 27, pp. 129--146, 1976.
- [ROB94] S. E. Robertson, S. Walker. Some Simple Effective Approximation of the 2-Poisson Model for Probabilistic Weighted Retrieval. *In ACM SIGIR*, pp.232--241, 1994.
- [ROC71] J. Rocchio. Relevance Feedback in Information Retrieval. *G. Salton édition, The SMART Retrieval System – Experiments in Automatic Document Proceesing*. Chapitre 14, pp. 313--323, Prentice Hall Inc.
- [SAG87] N. Sager, C. Friedman, M. Lyman. Medical Language Processing : Computer Management of Narrative Data. *Addison Wesley*, 1987.
- [SAH98] M. Sahami, Using Machine Learning to Improve Information Access, *Thèse de doctorat*, Université de Stanford, Computer Science Department, 1998.

- [SAL70] G. Salton. Automatic Text analysis, *Science*, N° 168, pp. 335--343, 1970.
- [SAL71] G. Salton. The SMART Retrieval System-Experiments in Automatic Document Processing. *Prentice Hall*, Englewood, Cliffs, New Jersey, 1971.
- [SAL88] G. Salton, C. Buckley. Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, Vol 24, N° 5, pp. 513--523, 1988.
- [SAL90] G. Salton, C. Buckley. Improving Retrieval Performance by Relevance Feedback. *Journal of the ASIS*, Vol. 41, N° 4, pp. 288-297, 1990.
- [SAR75] T. Saracevic. RELEVANCE : A Review and a Framework for the Thinking on the Notion in Information Science. *Journal of the ASIS*. Vol. 26, N° 6, pp. 321--343, 1975.
- [SCH92] P. Schäuble, D. Knaus. The Various Roles of Information Structures. *In Information and Classification*, pp. 282--290, 1992.
- [SCS95] H. Schuetze, D. Hull, J. Pedersen. A comparison of document representations and classifiers for the routing problem. *In Proceedings of the 18<sup>th</sup> Annual ACM SIGIR Conference*. pp. 229--237, 1995.
- [SCH94] H. Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. *In Proceedings of International Conference on New Methods in Language Processing*, 1994.
- [SIN96] A. Singhal, C. Buckley, M. Mitra. Pivoted Document Length Normalization. *In ACM SIGIR*, pp. 21-29, 1996.
- [SpJ73] K. Sparck Jones, M. Kay. Linguistics and Information Science. *Academic Press*, New York, 1973.
- [SOD95] W. Soderland, D. Fisher, J. Aseltine, W. Lehnert. CRYSTAL : Inducing a conceptual dictionary. *In Proceedings of International Joint Conference on Artificial Intelligence*, pp. 1314--1319, 1995.
- [TREC] <http://trec.nist.gov/>
- [TUR91] H. Turtle, W.B. Croft. Efficient Probabilistic Inference for Text Retrieval. *Proceedings of RIAO 3*, 1991.
- [VaR79] C. J. Van Rijsbergen. Information Retrieval. *Butterworths*, Londres, 1979. (2<sup>ème</sup> édition).
- [WIE95] E. Wiener, J. O. Pedersen, A. S. Weigend. A Neural Network Approach to Topic Spotting. *In Symposium on Document Analysis and Information Retrieval*, pp. 317--332, 1995.





# Chapitre 5

## Extraction de passages

**Résumé.** Dans ce chapitre nous introduisons les problématiques de la segmentation et du résumé de texte. Pour le problème de segmentation nous décrivons deux grandes familles de méthodes : les techniques qui fonctionnent à partir du calcul de similarités et celles basées sur des notions de cohésion. Ces techniques de segmentation sont utilisées en particulier pour la constitution de résumés. Nous introduisons ensuite les méthodes de résumé de texte et passons en revue les grandes familles de techniques utilisées.

**Mots clés :** Extraction de passages, segmentation de texte, résumé de texte.

### 5.1 Introduction

Avec l'augmentation de la taille des documents dans les collections de textes, la recherche et l'extraction de passages pertinents connaissent un essor particulier depuis une dizaine d'années. Les documents longs ou ceux possédant une structure complexe ou même les documents courts contenant plusieurs thèmes sont un défi pour les algorithmes qui ne distinguent pas quelle partie d'un document s'apparie avec une requête donnée. Les études dans ce domaine montrent que pour les documents ayant une structure hiérarchique interne, la recherche sur les passages peut améliorer les résultats de la recherche documentaire [SAL91, SAL93, MOF94, CAL94, MITT94]. Pour les documents structurés avec une hiérarchie de sections, de paragraphes et de phrases, chaque élément de cette structure est une source d'évidence qui peut être utilisée dans la recherche. Les sections et les paragraphes sont considérés comme des entités *globales* tandis que les phrases sont vues comme des entités *locales*. La combinaison des entités aux différents niveaux de la hiérarchie peut donner une meilleure performance en recherche documentaire que dans le cas où on considère seulement le document en entier [WIL94].

Une façon de traiter ce problème est l'approche basée sur l'extraction de passages proposée par [HEA93, MOF94]. Au moment de l'indexation, un document est subdivisé en passages. Chaque passage est ainsi considéré comme une entité distincte. Cette

approche a la mérite d'être simple, car il n'y a pas besoin de nouveaux algorithmes. Par contre, la façon de subdiviser un document en différents passages reste problématique [SAL93]. Cette méthode soulève également une autre difficulté : les passages courts vont avoir peu de termes en commun avec les requêtes et ceci bien qu'ils puissent être pertinents.

Un autre exemple est le travail de O'Connor et de Ro [OCO80, RO88]. Ils montrent que si chaque portion de texte ou *passage* est triée dans l'ordre décroissant de leur pertinence par rapport à une requête, il est alors simple de mettre en place une interface Homme-Machine pour aider l'utilisateur à trouver l'information qu'il recherche. Ces études ne tiennent pas compte de la longueur des documents et elles ont été évaluées sur des bases de documents de taille petite ou moyenne.

L'extraction de passages est à la base de nombreuses méthodes de segmentation (section 2) ou de résumé de textes (section 3) qui sont les deux tâches auxquelles nous allons nous intéresser dans ce chapitre. Ce chapitre nous servira à positionner le travail présenté dans le chapitre 7 qui est consacré au résumé.

## 5.2 Segmentation de Texte

### 5.2.1 Introduction

La tâche de segmentation consiste à identifier des régions de texte possédant certaines propriétés. Découper un texte, en passages cohérents par exemple, apporte une information substantielle pour de nombreuses applications en RI [KLA98]. C'est cette problématique dont nous allons traiter ici. De nombreuses études sur le sujet montrent qu'une telle tâche, qui semble pourtant être simple, s'avère difficile à automatiser.

Citons quelques exemples montrant l'intérêt de la segmentation.

Dans le cadre de la recherche documentaire, rapporter de longs textes en réponse à une requête n'est pas forcément une bonne solution car l'utilisateur doit examiner beaucoup de données [SAL96]. La segmentation de texte permet aussi de structurer les documents suivant leurs thèmes. Ceci est d'autant plus important que plusieurs expériences montrent que les méthodes d'indexation généralement employées par les systèmes de recherche documentaire masquent certaines thématiques des documents. Ainsi certains documents pertinents n'apparaissent pas dans les (premières) réponses fournies par le système car la portion de texte traitant de la même thématique que la requête n'a pu être isolée.

### 5.2.2 Quelques algorithmes pour la segmentation

Dans ce qui suit nous allons présenter quelques algorithmes classiques pour la segmentation de textes. Ces algorithmes cherchent à isoler sur différents types de passages de textes, qui peuvent se classer en trois catégories [CAL94] :

- A) les segments de *discours* qui sont basés sur des unités textuelles de discours (comme les phrases, les paragraphes et les sections),
- B) les segments *sémantiques* qui sont basés sur un sujet ou sur le contenu du texte (comme le *text tiling* [HEA97]),
- C) les segments *contextuels* qui sont basés sur le nombre de mots qu'ils contiennent.

Nous distinguerons deux grandes familles de méthodes celles qui reposent sur une mesure de similarité métrique entre passages successifs et celles qui reposent sur des chaînes lexicales.

### 5.2.2.1 Segmentation par calcul de la similarité

Beaucoup de techniques sur l'extraction de passages recherchent des entités textuelles de taille fixe [CAL94] ou des entités définies par la structure du document, comme les paragraphes [SAL93]. Mais dans la plupart des cas, l'information sur les paragraphes ou les sections n'est toujours pas disponible et les segments de textes sont souvent constitués d'un petit nombre de phrases, et même parfois d'une seule phrase. Choisir ainsi une taille arbitraire fixe pour les segments de textes n'est pas approprié à la tâche de segmentation.

#### *Segmentation thématique*

La plupart des algorithmes de segmentation thématique font l'hypothèse que si les représentations vectorielles de deux extraits ont une faible similarité alors ces extraits ont de faibles liens thématiques. D'après cette hypothèse, deux extraits peu similaires (au sens d'une mesure donnée) donneront lieu à une segmentation du document qui les contient. Salton et al. [SAL94a, SAL96] discutent de la décomposition de textes en segments et en thèmes où un segment est un bloc continu du texte parlant d'un seul sujet et un thème est une chaîne de tels segments. Dans leur approche, le processus de segmentation commence au niveau des paragraphes. Les paragraphes sont ensuite comparés entre eux en calculant une mesure de similarité. Une extension possible de cette méthode consiste en l'utilisation de deux niveaux de granularité. Au premier niveau, il s'agit de détecter les zones de rupture les plus probables (en fonction des similarités entre les segments adjacents). Au second niveau il s'agit de déplacer les frontières de phrase en phrase jusqu'à minimiser les similarités entre les segments qu'elles séparent.

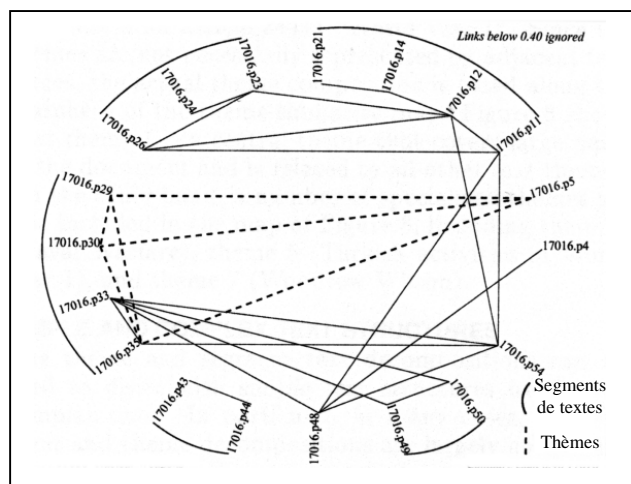


Figure 28. Graphe de segmentation et des liens thématiques d'un document. Chaque paragraphe est représenté par un numéro, ceux-ci sont positionnés dans l'ordre d'apparition dans le texte. Seuls sont reliés les paragraphes dont la similarité est supérieure à un seuil [SAL96].

[SAL94c] proposent une stratégie de segmentation mixte (*global-local text comparison*) calculant des similarités entre des zones de textes étendues (les paragraphes par exemple), puis, si ces valeurs sont inférieures à un seuil, entre les phrases de ces zones. [SAL94d] utilisent cette technique en ne calculant pas seulement les similarités de paragraphes adjacents mais celles de tous les paragraphes pris deux à deux (figure 28). Cette méthode permet de détecter les segments de textes se détachant des autres segments [SAL96].

### ***Text tiling***

Hearst et al. [HEA93, HEA94, HEA97] présentent une méthode de la segmentation de textes en sous-thèmes qu'ils appellent *tiling*.

L'entité de base pour cet algorithme est le segment de texte qui est défini par un nombre fixe de phrases le constituant. L'algorithme de *Text tiling* calcule un score pour chacun des segments de texte en fonction du segment qui le suit. Ce score dépend d'un autre score attribué à chaque paire de phrases, contenue dans le segment (*lexical score*). Le score pour chaque paire de phrases est calculé en tenant compte de l'un des critères suivants :

- Les mots communs comptabilisés par le produit scalaire entre les vecteurs représentatifs des deux phrases,
- Le nombre de mots nouveaux dans ces phrases,
- Le nombre de chaînes lexicales actives (une chaîne lexicale relie les occurrences de termes sémantiquement proches et elle est dite active si la distance qui sépare l'occurrence d'un terme avec son occurrence suivante est inférieure à un seuil donné et que la paire de phrases en cours d'analyse est incluse dans cette chaîne au moins en partie).

Ainsi le score d'un segment de texte est le produit normalisé des scores de chaque paire de phrases qu'il contient (*lexical scores*). Si l'écart entre le score d'un segment et les scores de celui qui le précède et de celui qui le suit est grand, une frontière thématique est définie pour ce segment. La rupture entre deux unités documentaires est située dans une zone du texte entourée de zones présentant des valeurs de cohésion très différentes de la sienne. Ces ruptures sont visibles sur le graphe des valeurs de cohésion par des creux ou des crêtes, cf. Figure 29. [HEA94] évalue les résultats de cet algorithme, et il en conclut qu'ils sont très similaires à ceux obtenus par des experts humains.

[HEA93] rapportent une amélioration des résultats de la recherche documentaire, après segmentation de l'un des corpus de TREC. Par contre, il n'a pas été montré qu'en général, pour la recherche documentaire, cette stratégie de segmentation donne de meilleurs résultats qu'une autre méthode.

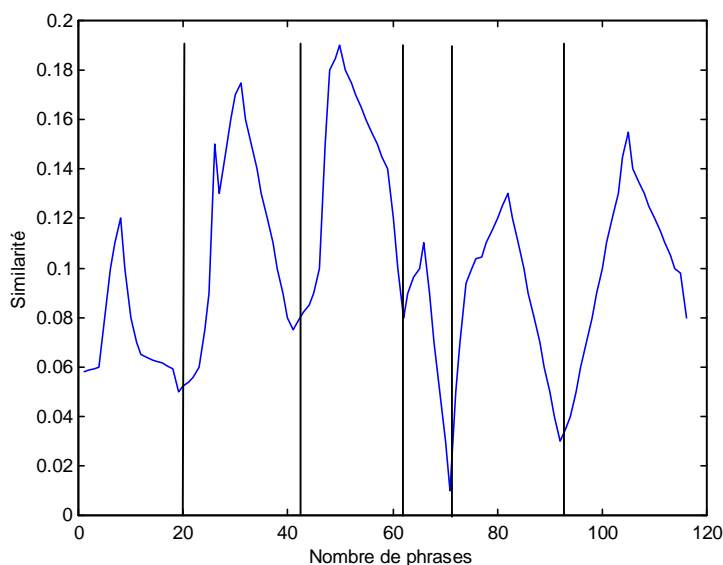


Figure 29. Méthode de *TextTiling*, le graphe indique un score entre segments adjacents du texte et les lignes verticales indiquent les frontières entre ces segments.

### Segmentation séquentielle

Mittendorf et Schäuble [MITT94] considèrent des passages « *sémantiques* » qui sont déterminés via un Modèle de Markov caché. Cette approche permet de tenir compte, dans le calcul de la pertinence d'un document par rapport à une requête, de la position des termes dans ce document (ce qui n'est pas habituellement le cas). De plus, elle résout en partie le problème de la pondération de termes grâce à l'utilisation de l'algorithme de Baum-Welch.

La figure 30 décrit ce modèle. La segmentation est faite grâce à une requête. Un texte est supposé être découpé en trois zones : une première partie non pertinente par rapport à la requête, une seconde partie pertinente et une troisième à nouveau non pertinente. Le modèle génère des extraits de texte pertinents en traversant l'état *R* et des extraits non pertinents en traversant l'état *I*. Pour un document donné, ce modèle permet de trouver le passage pertinent par rapport à la requête d'utilisateur.

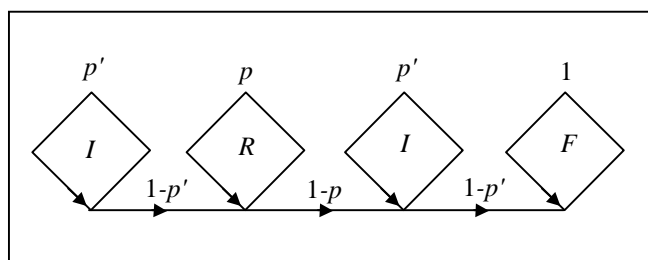


Figure 30. Modèle de Markov caché pour la segmentation de texte.

La zone pertinente est supprimée du document et le texte restant dans le document est de nouveau soumis au modèle de Markov et une nouvelle zone de pertinence maximale en est extraite. Ce processus est réitéré suivant les désirs de l'utilisateur auquel est retournée une liste de segments ordonnés suivant leur pertinence. De cette façon on en déduit une segmentation dont les frontières sont celles qui séparent les segments extraits. Ce modèle n'a pas véritablement été testé sur une tâche qui permette de juger de sa pertinence, il constitue malgré tout une approche originale au problème.

#### 5.2.2.2 Cohésion lexicale

##### *Cohésion lexicale*

[HAL76, KAL88, WAL91] étudient la répétition de termes dans les phrases comme indicateur de cohésion. Ce facteur de répétition a une importance variable suivant les différents types de texte étudiés. Il existe des textes (comme les articles scientifiques) ayant un vocabulaire spécifique et où le nombre d'homonymes est souvent réduit [HEA97]. A l'opposé, il y a des textes où le nombre de termes, exprimant une même notion, est souvent élevé et il n'est pas un bon indicateur de cohésion.

Kan et Klavans [KAN98], font une segmentation linéaire en étudiant les chaînes lexicales présentes dans le texte cible [MOR91]. Ces chaînes relient les occurrences d'un terme dans les phrases des documents à segmenter. Une chaîne est rompue si le nombre de phrases séparant deux occurrences est trop important. Ce nombre dépend de la catégorie syntaxique du terme considéré. Par exemple, un nom propre dans un texte donné désigne généralement toujours la même personne mais il est souvent remplacé par un pronom lui faisant référence. Ainsi le nombre maximal de phrases séparant deux occurrences d'un nom propre peut être élevé sans que la thématique du texte n'ait changé. Ce nombre est en général plus faible pour les noms communs (4 phrases) puisqu'un même nom commun peut avoir plusieurs sens [KAN98].

Une fois toutes les chaînes établies, un poids leur est assigné en fonction de la catégorie syntaxique des termes en jeu et de la longueur de la chaîne. Un score est ensuite donné à chaque paragraphe en fonction des poids et des origines des chaînes qui le traversent. Des marques de segmentation sont apposées au début des paragraphes ayant les scores maximaux. Cet algorithme de segmentation est évalué sur 15 articles du *Wall Street Journal* et 5 de *Economist* [KAN98]. La taille de ces bases pose un problème quant à la validité des résultats, mais le coût pour la constituer (chaque document est traité par un expert humain) ne laisse guère le choix. Sur cette base, leur algorithme donne de meilleurs résultats que le *Text Tiling* aussi bien en terme de précision qu'en terme de rappel des segments trouvés. [KAN98] utilisent la segmentation obtenue pour détecter les segments les plus représentatifs des textes et créer automatiquement un résumé.

### ***Cohésion lexicale étendue***

Dans le cas où un concept serait exprimé par des termes différents, l'analyse de la répétition des termes n'est plus suffisante pour détecter les marques de transition thématique. [MOR91, KOZ93] proposent de calculer un coefficient de cohésion lexicale (*Lexical Cohesion Profile*) pour chaque segment du texte en fonction de la ressemblance des mots présents.

La cohésion entre deux termes est évaluée d'après un thesaurus (*Roget's thesaurus*) [MOR91] ou un réseau sémantique construit d'après un dictionnaire [KOZ93]. Elle correspond au calcul d'une similarité en fonction de la proximité sémantique de chacun des deux termes et de leur usage commun dans les textes à segmenter. La cohésion d'un segment de texte dépend de la similarité des termes le constituant. Une segmentation thématique peut alors être réalisée en marquant les zones de faible cohésion.

[FER98] proposent de construire automatiquement un réseau sémantique de collocations – sur un corpus d'entraînement et non d'après les définitions d'un dictionnaire comme [KOZ93], pour déterminer la cohésion entre les termes suivant la mesure d'information mutuelle définie par [CHC90].

Pour chaque position d'une fenêtre dans le texte, [FER98] et [KOZ93] calculent une valeur de cohésion de cette fenêtre en fonction du poids des mots dans la fenêtre et du poids des mots du réseau de collocations contenant au moins deux termes communs à ceux de la fenêtre. Le déplacement de la fenêtre permet de distinguer les zones de forte et de faible cohésion. Une marque de segmentation est déposée à la position correspondant à une forte baisse de la valeur de cohésion (transition entre deux zones de forte cohésion).

Les résultats des algorithmes présentés par [MOR91, KOZ93, FER98] n'ont pas été comparés sur des corpus identiques. Ils n'ont pas été appliqués à la recherche documentaire.

Ponte et Croft [PON97] présentent une approche différente: ils utilisent la phrase comme entité textuelle de base. En commençant par une phrase, ils l'enrichissent d'abord en utilisant la méthode LCA (*Local Context Analysis*) [XU96] qui est une méthode d'enrichissement de requête basée sur le *retour utilisateur*. En utilisant des techniques de programmation dynamique, ils déterminent ensuite les phrases à la droite de la phrase enrichie qui lui sont sémantiquement proches.

La plupart de ces travaux extraient des passages de tailles fixes. Une exception est le travail de Mittendorf et Schäuble [MITT94]. Dans leurs travaux l'information est modélisée par un processus stochastique qui génère des fragments de textes pertinents par rapport à une requête donnée. Cette procédure est appelée *modèle de passage* ou *segmentation par requête* par opposition aux approches comme celle de [PON97] où le texte est segmenté par thème, indépendamment d'une requête donnée (*modèle générique de segmentation*).

#### **5.2.2.3 D'autres approches pour la segmentation**

De nombreuses autres approches ont été proposées. Elles reposent sur l'utilisation de classifieurs ou d'algorithmes ad-hoc.

Litman et Passoneau [LIT95] utilisent des arbres de décision pour la segmentation en s'appuyant sur certaines caractéristiques linguistiques déduites de corpus oraux.

Dans Bigi et al. [BIG98] un certain nombre de thématiques sont apprises sur une base d'apprentissage. Ils emploient alors un modèle de langage pour étiqueter chaque paragraphe d'un texte suivant sa thématique de plus forte probabilité. Lorsque la valeur de la probabilité du meilleur thème décroît, une frontière thématique possible est détectée. La sélection définitive des frontières est effectuée suivant une méthode de programmation dynamique.

Yamron et al. [YAM98, VaM98] présentent une méthode de segmentation basée sur les MMC. Les états cachés du MMC représentent des thèmes prédéfinis et les observations sont des mots ou des phrases. Ils divisent d'abord le texte en régions sémantiquement proche en utilisant l'algorithme des *K-moyennes*. La distance utilisée dans cet algorithme est la distance de *Kullback*. Les transitions entre les états sont ensuite déterminées à la main sur un petit nombre de thématiques analysées par des experts.

Beeferman et al. [BEE97, BEE99] construisent un modèle exponentiel qui à chaque phrase fait correspondre la probabilité qu'il y ait une frontière entre cette phrase et la phrase suivante. La distribution de probabilité est choisie en construisant de façon incrémentale un modèle log-linéaire.

### 5.2.3 Evaluation des systèmes de segmentation

En général, il existe deux méthodes pour l'évaluation des systèmes de segmentation. La première consiste en une comparaison de la décision de ces systèmes avec des jugements humains : aucun corpus segmenté de taille suffisante n'est cependant disponible à ce jour ; des propositions ont été faites pour la constitution d'un tel corpus [NAK95] et pour évaluer la qualité des jugements humains [LIT95, HEA97].

La deuxième consiste à comparer le résultat de la segmentation à des marques déposées par un unique lecteur. Cette procédure d'évaluation est bien évidemment subjective.

## 5.3 Résumé de Texte

### 5.3.1 Introduction

Les systèmes classiques de *RI* renvoient à l'utilisateur une liste ordonnée des documents les plus pertinents par rapport à sa requête. Mais, tous les documents retournés ne sont pas forcément pertinents, et leur examen est coûteux en temps. Présenter à l'utilisateur des résumés de documents peut grandement l'aider dans sa recherche (tâche ad-hoc). Un résumé peut aussi aider l'utilisateur à catégoriser plus facilement les documents (tâche de catégorisation) ou à répondre à des questions (tâche question-réponse).

Les résumés similaires à ceux réalisées par un humain (résumé manuel) sont néanmoins difficiles à faire sans une compréhension poussée du contenu du texte [SpJ93]. Il existe beaucoup trop de variation de styles d'écriture, de constructions syntaxiques, etc., pour pouvoir construire un système de résumé générique. Un système idéal de résumé comprend l'information pertinente que l'utilisateur cherche et en exclut les informations



redondantes. Ces informations doivent être cohérentes et compréhensibles, toutes choses qui sont difficiles à faire sans utiliser des processus du langage naturel.

Pour contourner ce problème, les systèmes de résumés proposent d'extraire des passages du texte, et présentent à l'utilisateur un résumé en concaténant ces passages. Il existe deux façons d'envisager le résumé automatique de texte :

- A) le *résumé générique* qui résume le contenu par rapport à l'idée principale du texte,
- B) le *résumé par rapport à une requête* qui résume le texte par rapport à une requête d'utilisateur.

Le résumé automatique de textes remonte à la fin des années cinquante aux travaux de Luhn [LUH58]. Pour extraire les phrases pertinentes nécessaires à la construction d'un résumé, il considère des caractéristiques comme la moyenne des fréquences de termes, des mots de titres et la position de la phrase. D'autres recherches sur ce problème utilisant des approches similaires ont continué jusqu'à la fin des années quatre-vingt [RAT61, EDM69, SKO72, TAI83, PAI90]. Avec l'avènement de l'Internet et de moteurs de recherche de plus en plus performants, l'importance d'informations synthétique du type résumé a considérablement augmenté et la tâche de résumé automatique a suscité de nouvelles vocations. Beaucoup de nouvelles approches ont commencé à être explorées :

- A) des approches qui tentent d'utiliser la structure du discours [MAR97b],
- B) des approches linguistiques [KLA95, McK95, AON97, BAL98, RAD98],
- C) des approches statistiques [CAR98, HOV97, MIT97]
- D) une combinaison de ces deux dernières approches (B et C) [BAR97, TEU98, Gol99a, MiK99].

Tous ces travaux sur le résumé, exceptés [KLA95, McK95] se sont intéressés au résumé de texte par extraction d'entités textuelles. Ces entités peuvent être des groupes de mots [BOU97], des clauses [MAR99a], des phrases [KUP95, TEU97, Gol99a, MiK99] ou des paragraphes [SAL94b, MIT97].

Ces techniques génèrent des résumés en concaténant des entités de base sélectionnées à partir du document original. Dans le cas général, le résumé de texte demande une compréhension, une interprétation, une abstraction de ce texte puis la génération d'un nouveau texte. Les procédés, vus plus haut, abordent cette tâche d'une manière beaucoup plus simple : ils recherchent l'ensemble ordonné d'entités du document original qui est le plus susceptible de faire parti du résumé.

### 5.3.2 Quelques algorithmes pour le résumé automatique de texte

Nous allons dans ce qui suit nous intéresser aux algorithmes les plus classiques pour le résumé de texte. La majorité des algorithmes existants s'appuient sur une mesure de similarité calculée entre les entités de base dans le texte (des paragraphes et des phrases)

[ZEC95, Gol99a]. Depuis quelques années, on assiste cependant à l'émergence de nouvelles techniques issues de l'apprentissage pour traiter ce problème.

### 5.3.2.1 Résumé basé sur des mesures de similarité

Le calcul de la similarité entre des entités textuelles peut se faire comme en recherche documentaire, i.e. par le biais d'une requête utilisateur ; ou encore génériquement, à partir d'une information tirée du corpus. On peut dans ce dernier cas ramener le problème à celui du résumé avec une requête, cette requête étant alors créée par exemple en utilisant les mots les plus fréquents dans le corpus d'apprentissage.

On adopte en général une approche vectorielle basée sur la projection des mots de la requête  $q$  sur chacune des phrases du texte, pour calculer le score entre cette requête et les phrases du document. Dans la majorité des algorithmes existants, chaque phrase  $\varphi_i$  est pondérée suivant la formule suivante :

$$Poids(\varphi_i) = \sum_{m \in \varphi_i} w_m \times (q \cdot \varphi_i)$$

où  $(q \cdot \varphi_i)$  est le produit scalaire entre la représentation vectorielle de la requête et celle de la phrase  $\varphi_i$  et  $w_m$  est le poids du mot  $m$  dans la base de documents. Les phrases sont alors triées dans l'ordre décroissant de leur score. Pour les autres études basées sur cette approche, les auteurs ajoutent des mesures sur certaines caractéristiques linguistiques [AON97, BAR97, Gol99a] qu'ils intègrent dans la formule ci-dessus; cela peut être par exemple, la prise en compte de noms propres ou l'utilisation de méthodes d'expansion basées sur un thesaurus (e.g. WordNet).

Une fois les scores calculés pour chaque phrase d'un document, un seuil qui tient compte de l'ensemble des scores est déterminé. Ce seuil permet de prendre une décision sur l'appartenance d'une phrase au résumé, figure 31.

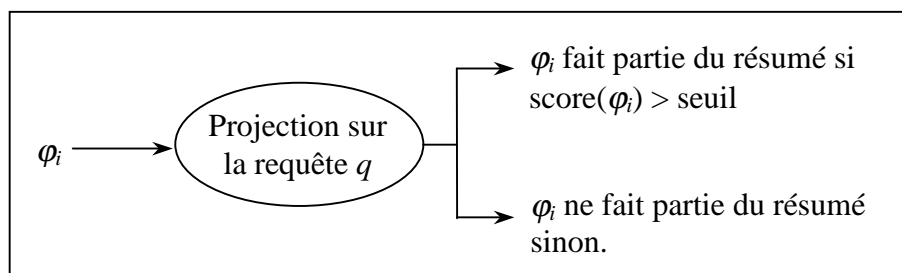


Figure 31. Schéma général d'un système de résumé automatique suivant l'approche « mesures de similarité ».

### 5.3.2.2 Résumé en respectant la structure rhétorique

Marcu [MAR97a] propose d'exploiter la structure rhétorique du texte pour engendrer des résumés automatiques (figure 32). Il utilise une théorie sur la structure rhétorique de Mann et Thompson [MaT87] qui postule qu'une phrase peut être décomposée en clauses.

D'après [MaT87] une phrase complexe est composée de deux parties ou clauses. Le segment principal est appelé *nucleus* et le segment subordonné est appelé *satellite*. Ce dernier est connecté au segment principal via une relation rhétorique. La figure 32 illustre deux exemples anglais de relations rhétoriques.

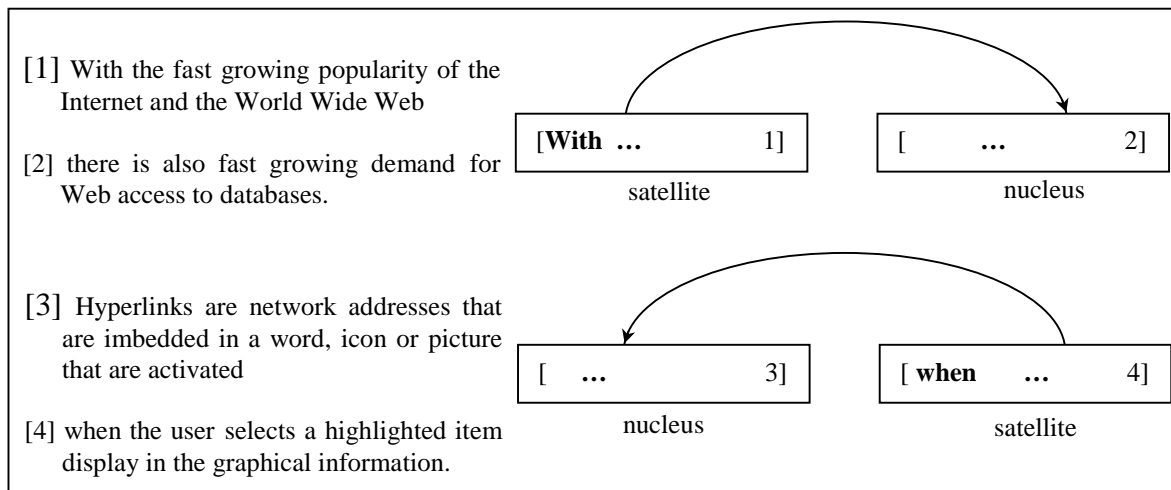


Figure 32. Les segments de phrases et leur relation rhétorique.

En général, lorsqu'une relation rhétorique existe, le nucleus est considéré être le segment principal et de ce fait il a plus de chance d'apparaître dans le résumé que le satellite.

Marcu [MAR97a] se base sur cette théorie pour créer la structure rhétorique de documents la « *Rhetorical Structure Theory Tree* ».

Il s'agit d'une structure d'arbre binaire qui est construite en tenant compte des relations nucleus-satellite : les segments les plus importants résident sur les feuilles supérieures et les segments les moins importants occupent les niveaux plus bas (cf. figure 33). Le résumé de texte est alors obtenu en coupant l'arbre à différents niveaux.

Un des inconvénients majeurs de cette approche est sa complexité de mise en œuvre. Ainsi pour chaque nouveau document sa structure rhétorique, qui est très coûteuse au niveau du temps de calcul, doit être extraite. De plus cette approche est très sensible au bruit dans la base de donnée, e.g. base hétérogène au niveau thématique, etc.

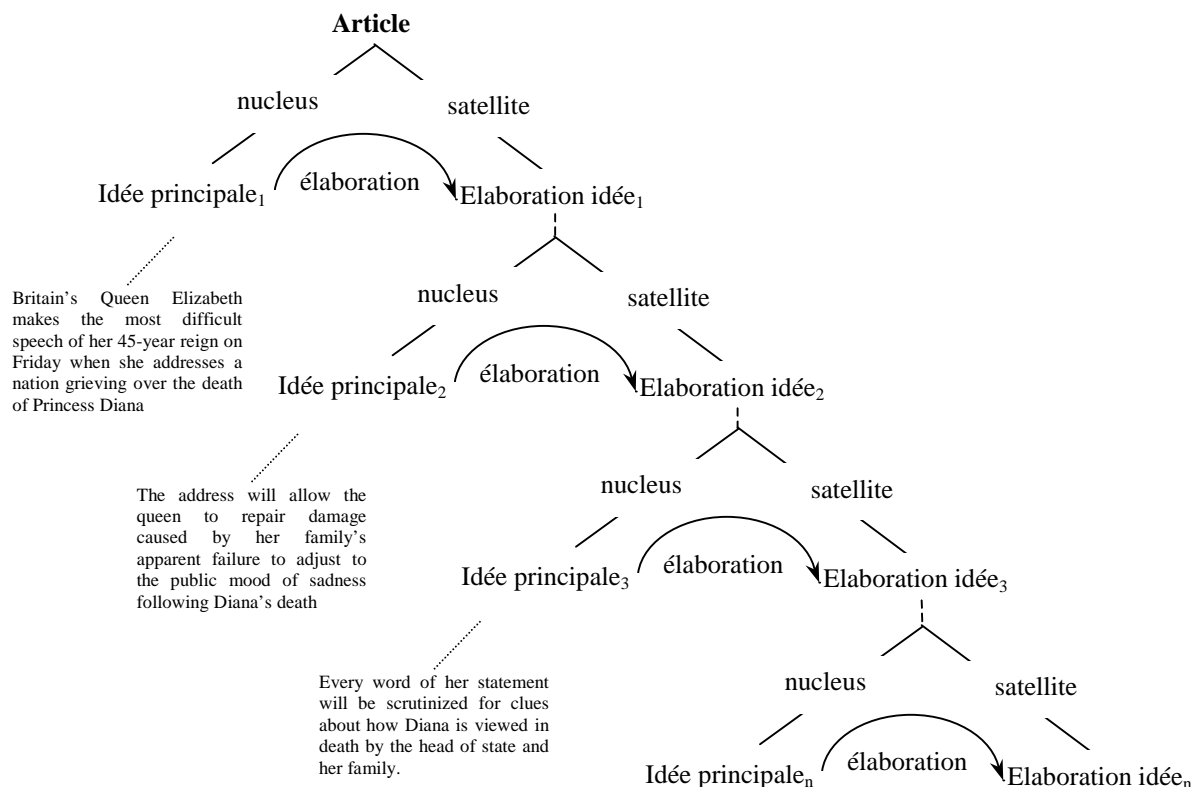


Figure 33. La structure rhétorique du discours.

### 5.3.2.3 Résumé par des méthodes d'apprentissage

Depuis 5 ans, des travaux sur le résumé ont commencé à intégrer l'utilisation de techniques issues de l'apprentissage. Dans ce qui suit, nous allons illustrer ces approches en nous intéressant aux deux méthodes d'apprentissage les plus répandues, basées respectivement sur la classification statistique et la décomposition de textes.

#### *Résumé à base de classification statistique de phrases*

[KUP95] et [TEU97, TEU98] considèrent le problème de l'extraction de phrases comme une tâche de classification statistique. Pour mettre au point leur fonction de classification, qui estime la probabilité qu'une phrase soit incluse dans le résumé extrait, ils utilisent comme base d'apprentissage, un ensemble de documents avec des résumés extraits associés.

Ils considèrent alors des caractéristiques  $F_j$ ,  $j=1, \dots, k$  qui sont caractéristiques du document comme la fréquence des mots clés, des mots de titres, la position des mots, ou caractéristiques des phrases comme la position de la phrase dans le document et sa longueur. [KUP95] définit à partir de ces caractéristiques un ensemble de critères que doit vérifier une phrase pour figurer dans le résumé. En utilisant une base d'apprentissage et la règle de Bayes, il estime ensuite pour chaque phrase  $\varphi$  la probabilité qu'elle soit dans le résumé  $S$  étant donné ses caractéristiques  $\{ F_j \}$ .

$$p(\varphi \in S / F_1, \dots, F_k) = \frac{p(F_1, \dots, F_k / \varphi \in S) \cdot p(\varphi \in S)}{P(F_1, \dots, F_k)}$$

En faisant l'hypothèse d'indépendance statistique entre les caractéristiques cette probabilité a posteriori s'écrit :

$$p(\varphi \in S / F_1, \dots, F_k) = \frac{\prod_{j=1}^k p(F_j / \varphi \in S) \cdot p(\varphi \in S)}{\prod_{j=1}^k p(F_j)} \quad (5.1)$$

$p(F_j / \varphi \in S)$  et  $p(F_j)$  sont estimés par la fréquence d'occurrence de la caractéristique  $F_j$  dans les phrases de la base d'apprentissage. Une fois l'apprentissage terminé, ils génèrent à partir d'un document un résumé en triant ses phrases suivant la probabilité donnée par la formule (5.1).

### **Résumé par décomposition de documents**

Jing et McKeown [JIN99] proposent l'utilisation de MMC pour apprendre à générer automatiquement des phrases de résumés écrites par des experts. Elles essaient de déterminer pour cela les relations entre les groupes de mots présents dans un résumé écrit manuellement et ceux présents dans un document. Cette idée repose sur l'idée – surement discutable- que les résumés manuels peuvent s'obtenir par un mécanisme de couper-coller à partir des textes du document initial. Cette approche est originale du fait qu'elle ne constitue plus un résumé à partir de la concaténation des entités textuelles fixes comme celles étudiées précédemment mais qu'elle cherche à créer un résumé en « collant » des groupes de mots à l'intérieur du texte.

Pour réaliser cela, elles utilisent un MMC. Elles disposent d'une base d'apprentissage dans laquelle les documents sont accompagnés de leur résumé manuel. Elles entraînent alors le modèle sur cette base. Pour cela elles considèrent chaque phrase du résumé comme une séquence de mots  $(w_1, \dots, w_N)$  où  $w_1$  est le premier mot de la phrase de résumé et  $w_N$  le dernier. La position d'un mot dans un document est identifiée uniquement par la position de la phrase et la position du mot dans la phrase ( $N^\circ$  phrase,  $N^\circ$  mot). Les occurrences multiples d'un mot dans le document peuvent alors être représentées par une liste de position de mots.

Elles font alors l'hypothèse que les mots dans la séquence de résumé manuel apparaissent suivant un modèle de bigramme : ainsi la probabilité d'apparition d'un mot à une position donnée dépend seulement du mot directement avant lui dans la séquence. Pour deux mots adjacents  $I_i$  et  $I_{i+1}$  dans le résumé manuel, elles estiment la probabilité  $p(I_{i+1} = (S_2, W_2) / I_i = (S_1, W_1))$  en utilisant la base d'apprentissage.

Ainsi pour chaque paire de mots contigus  $(I_i, I_{i+1})$  dans la séquence de résumé, elles regardent leurs positions spatiales (position de la phrase et position du mot dans la phrase) dans le document et en déduisent la probabilité  $p(I_{i+1} / I_i)$ . Elles utilisent l'algorithme de Viterbi pour déterminer le meilleur chemin. Ce modèle de MMC peut ensuite être appliqué pour faire le résumé générique d'un document quelconque.

Il est à noter que ce modèle bien qu'original, au point de vue mise en œuvre, pose quelque problèmes au niveau de son évaluation. En effet, il est difficile de voir comment la position spatiale des mots peut apporter l'information suffisante pour constituer un résumé de texte. [Jin99] évaluent leur système sur la base Ziff-Davis (associée à TREC7) constitué de dix documents et des résumés manuels associés. La bonne performance qu'elles obtiennent sur cette base ne garantit en aucun cas le bon fonctionnement de ce système dans le cas général.

### *D'autres algorithmes d'apprentissage pour le résumé de texte*

Barzilay et al. [BAR97] explorent l'utilisation des chaînes lexicales, comme dans les méthodes basées sur des mots clés statistiques, ils modélisent les concepts d'un document en utilisant des entités lexicales observées dans le document. Chuang et Yang [CHU00] définissent des caractéristiques structurées (relatives aux relations rhétoriques) et non-structurées (les mots de titres) pour construire une représentation vectorielle des segments de texte et appliquent des algorithmes d'apprentissage numérique comme ceux utilisés par [KUP95] et [TEU97] pour entraîner leur système de résumé.

### **5.3.3 Génération de résumés extraits à partir de résumés manuels**

Du fait de la variabilité des documents en terme de longueur, de style d'écriture et de lexique utilisé, les systèmes de résumé à base d'extraction de passages sont difficiles à mettre au point. Tout d'abord, peu de corpus de résumés existent actuellement, ce qui rend difficile l'évaluation des systèmes (cf. section 5.3.4). Il est très coûteux de générer manuellement de tels corpus.

En revanche, il est plus simple de trouver des résumés manuels pour un document que des passages extraits faisant office de résumé. De ce fait, il existe beaucoup plus de résumés, écrits manuellement que de résumés extraits. Pour tenir compte de ces ressources, de nouvelles techniques ont été proposées afin de transformer les résumés manuels en des résumés composés de passages pertinents. Ceci afin de constituer des bases d'apprentissage ou de test. Pour cela on apparie les phrases écrites dans les résumés manuels aux phrases du texte (figure 34).

Notons également que plusieurs chercheurs se sont intéressés au problème de l'alignement de passages pertinents des textes contenus dans les corpus parallèles et écrits en deux langues différentes [FUN94, WU94]. Banko et al. [BAN99] proposent une méthode basée sur le calcul de la longueur et de la fréquence des termes (*Term Length Term Frequency – TLTF*) pour déterminer la similarité entre les phrases des résumés manuels et des phrases du texte. Ainsi, pour chaque phrase du résumé, ils cherchent les phrases du document qui ont une grande mesure de similarité au sens *TLTF*. Ils construisent ainsi un résumé ne contenant que les phrases du document ayant la plus grande similarité avec les phrases du résumé manuel.

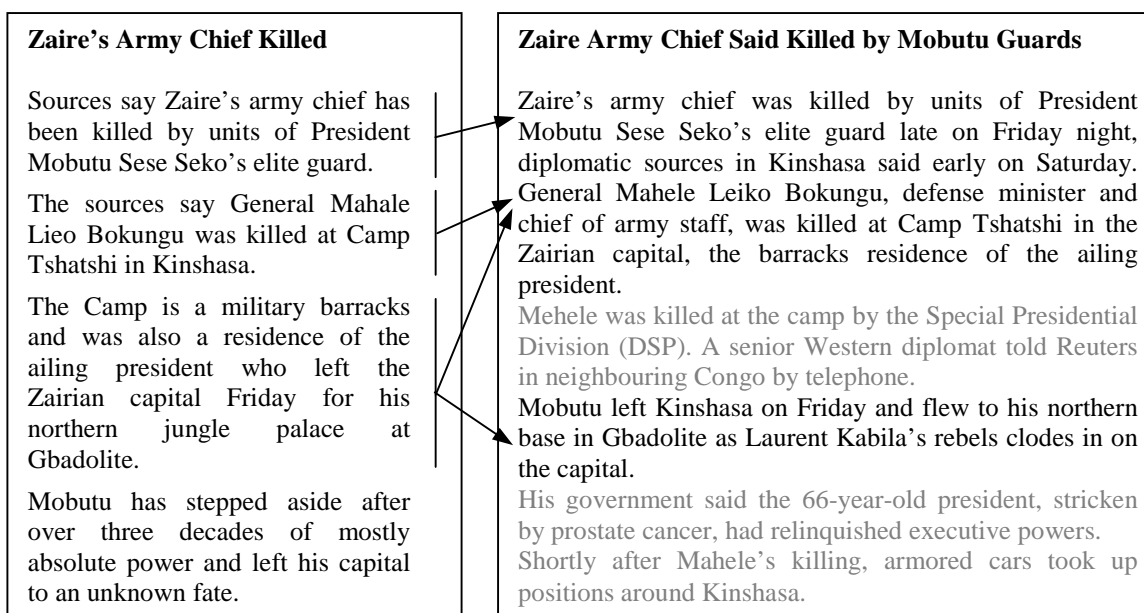


Figure 34. Un exemple de résumé manuel (gauche) et un document (droite). L'alignement des passages est montré sur la figure. Aucun passage du document ne correspondent à la dernière ligne du résumé.

Marcu [MAR99b] propose une approche itérative basée sur la similarité d'un ensemble de phrases avec le résumé de texte écrit manuellement. En commençant avec l'ensemble de toutes les phrases du document, il exclut une à une les phrases qui font croître la similarité de cet ensemble - lorsqu'on les retire de cet ensemble - avec le résumé manuel. Il arrête cette procédure dès qu'il n'y a plus de phrases vérifiant cette condition. L'ensemble des phrases restantes constitue alors un résumé du document dit « extract ». L'algorithme 16, résume cette procédure itérative.

- 
1. Filtrer le document et le résumé manuel à travers un ante-dictionnaire.
  2. Normaliser chaque mot suivant sa racine
  3. Construire un ensemble des phrases du document  $E_M = \{\text{Phrases du document}\}$ .
  4. Faire tant que Flag=1
 

S'il existe une phrase  $\phi_i$  tel que  $\forall j, j \neq i \text{ sim}(E_M | \phi_i) \geq \text{sim}(E_M | \phi_j)$

Flag=1

$E_M = E_M \setminus \phi_i$

Sinon

Flag=0
  5. Nettoyer l'ensemble  $E_M$  restant suivant des considérations linguistiques
  6. Poser le résumé extrait =  $E_M$
- 

Algorithme 16. L'Algorithme de construction de Marcu pour construire des résumés extraits à partir de résumés manuels.

On note que la similarité entre le résumé manuel et le nouvel ensemble de phrases obtenu augmente à chaque itération ceci jusqu'au moment où il ne reste plus de phrase liée au résumé manuel Figure 35.

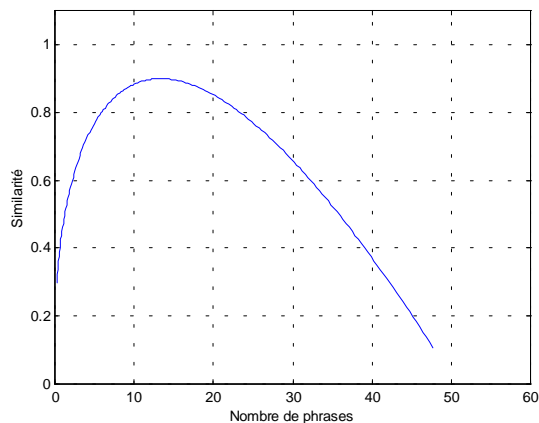


Figure 35. Comportement de l'algorithme d'extraction de résumé de Marcu

### 5.3.4 Evaluation et perspectives

#### *Evaluation*

L'évaluation des systèmes de résumés peut être intrinsèque ou extrinsèque [JON96]. Les méthodes intrinsèques essaient de mesurer la qualité d'un système, et les méthodes extrinsèques essaient de mesurer la performance d'un système pour une tâche donnée (on évalue sur une liste de phrases étiquetées pertinentes pour la tâche). La plupart des méthodes d'évaluation pour des systèmes de résumé sont intrinsèques : la qualité des résumés est jugée par rapport à un jugement humain. Jing et al [JIN98] donnent un aperçu des différentes méthodes d'évaluations.

Jusqu'à présent deux campagnes d'évaluations extrinsèques de systèmes de résumé ont été lancées : TIPSTER et SUMMAC [NIST93, SUM98]. SUMMAC est une version plus élaborée de TIPSTER. Cette dernière a été la première campagne d'évaluation de systèmes de résumé automatique à grande échelle. Les deux tâches principales pour l'évaluation extrinsèque dans SUMMAC sont :

- A) la tâche ad-hoc dont le but est de trouver les résumés relatifs à un sujet donné,
- B) la tâche de catégorisation dont la finalité est de trouver quand un résumé générique peut présenter l'information suffisante pour permettre à un analyste de catégoriser rapidement et correctement le document correspondant.

Dans cette campagne, il y a aussi la tâche de question-réponse qui implique une évaluation intrinsèque. Pour cette tâche un résumé dépendant d'un sujet donné est évalué par rapport aux réponses (trouvées dans le document source) qu'il contient et qui sont pertinentes pour un ensemble de questions relatives au sujet.

La conclusion de SUMMAC a été que les systèmes de résumé automatique donnent de bonnes performances en regard d'une évaluation extrinsèque. Ainsi les résumés avec un



taux de compression bas (de l'ordre de 17% pour la tâche ad-hoc et 10% pour la tâche de catégorisation) représente une dégradation minimale (5% de dégradation de la F-mesure pour la tâche ad-hoc et 14% de dégradation pour la tâche de catégorisation) dans le cas où on utilise ce résumé à la place du document en entier.

Néanmoins dans le cas de l'évaluation intrinsèque la plupart des études sur le résumé automatique considèrent cette évaluation comme une tâche difficile et proposent chacune une méthode différente. Ce genre d'évaluation individuelle rend toute comparaison entre les différents systèmes difficile et le problème de l'évaluation reste largement ouvert.

### *Perspectives*

Depuis peu, des techniques pour résumer simultanément plusieurs documents se développent. La présence de grandes quantités d'informations disponibles sur le WEB rend de plus en plus difficile l'accès à une information répartie sur un grand nombre de documents. Beaucoup de recherches ont essayé d'étendre les approches centrées sur le résumé de documents simples au résumé de documents multiples [MAN97, RAD98, Gol99b]. Ces techniques incluent la comparaison de champs de structures remplis grâce aux techniques d'extraction d'information et la génération de résumés en langage naturel à partir de ces champs [RAD98]. D'autres approches comparent différentes sections dans plusieurs documents et identifient celles qui ont des points communs [TIP98]. Mani et al. [MAN97] construisent des réseaux d'activation des entités lexicales et extraient des phrases de l'ensemble des documents.

## **5.4 Conclusion**

Dans ce chapitre nous avons présenté les deux tâches de segmentation et de résumé de texte. Ces tâches sont abordées sous l'angle de l'extraction de passages. Nous avons introduit ces tâches comme un état à l'art au chapitre 7 qui est consacré au résumé automatique de texte.

## 5.5 Bibliographie

- [AON97] C. Aone, M. E. Okurowski, J. Gorlinsky, B. Larsen. A Scalable Summarization system using Robust NLP. *In Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*. pp. 66--73, Espagne, 1997.
- [BAL98] B. Baldwin, T. S. Morton. Dynamic Coreference-based Summarization. *In Proceedings of the Third Conference on Empirical Methods in Natural Language Processing*, Espagne, 1998.
- [BAN99] M. Banko, V. Mittal, M. Kantrowitz, J. Goldstein. Generating Extraction-Based Summaries from Hand-Written Summaries by Aligning Text Spans. *In the Proceedings of the Pacific Association for Computational Linguistics*, 1999.
- [BAR97] R. Barzilay, M. Elhadad. Using Lexical Chains for Text Summarization. *In Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*. pp. 10--17, Espagne, 1997.
- [BEE97] D. Beeferman, A. Berger, J. Lafferty. Text Segmentation using Exponential Models. *In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pp. 35--46, 1997.
- [BEE99] D. Beeferman, A. Berger, J. Lafferty. Statistical Models for Text Segmentation. *Machine Learning special issue on Natural Language Learning*, C. Cardie and R. Mooney eds., pp. 177--210, 1999.
- [BIG98] B. Bigi, R. de Mori, M. El-Bèze, T. Spriet. Detecting topic shifts using a cache memory. *In the proceedings of the fifth conference on Spoken Language Processing*. 1998.
- [BOU97] B. Boguraev, C. Kennedy. Saliency-based content Characterization of Text Documents. *In Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*. pp. 2--9, Espagne, 1997.
- [CAR98] J. G. Carbonell, J. Goldstein. The use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. *In Proceedings of SIGIR*, 1998.
- [CAL94] J. P. Callan. Passage-Level Evidence in Document Retrieval. *In Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. pp. 302--310, Dublin, Ireland, Juillet 1994.
- [CHC90] K. W. Church, P. Hanks. Word association norms, mutual information and lexicography. *Computational Linguistics*, pp. 22--29, vol. 16, n°1, 1990.

- [CHU00] W. T. Chuang, J. Yang. Extracting Sentence Segments for Text Summarization: A Machine Learning Approach. *In Proceedings of the 23<sup>th</sup> ACM SIGIR Conference on Research and Development in Information Retrieval*. pp. 152--159, Athènes, Grèce, 2000.
- [EDM69] H. P. Edmundson. New Methods in Automatic Abstracting. *Journal of the ACM*. Vol. 16, N° 2., pp. 264--285, Avril 1969.
- [FER98] O. Ferret, B. Grau, N. Masson. Thematic segmentation of texts : two methods for two kinds of texts. *Actes of COLING-ACL*, pp. 392--396, 1998.
- [FUN94] P. Fung, K. Church, K. K-vec. A new approach for aligning parallel texts. *In Proceedings of COLING-94*, 1994.
- [Gol99a] J. Goldstein, M. Kantrowitz, V. Mittal, J. Carbonell. Summarizing Text Documents: Sentence Selection and Evaluation Metrics. *In Proceedings of the 22<sup>th</sup> ACM SIGIR Conference on Research and Development in Information Retrieval*. 1999.
- [Gol99b] J. Goldstein. Automatic Text Summarization of Multiple Documents. *Thesis Proposal*. Juin 1999.
- [HAL76] M. A. K. Halliday, R. Hasan. Cohesion in English. *Longman*.
- [HEA93] M. Hearst, C. Plaunt. Subtopic Structuring for full-length Document Access. *Proceedings of the sixteenth Annual International ACM SIGIR Conference*. pp. 59--68, Pittsburgh, PA, 1993.
- [HEA94] M. Hearst. Multi-Paragraph Segmentation of Expository Text. *Proceedings of the 32<sup>nd</sup> Annual Meeting of the Association for Computational Linguistics*. Las Cruces, 1994.
- [HEA97] M. Hearst. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, pp. 33--64, 1997.
- [HOV97] E. Hovy, C. Y. Lin. Automated Text Summarization in SUMMARIST. *In Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*. pp. 18--24, Espagne, 1997.
- [JIN98] H. Jing, R. Barzilay, K. McKeown, M. Elhadad. Summarization Evaluation Methods: Experiments and Analysis. *In Proceedings of the AAAI-98 spring Symposium on Intelligent Text Summarization*. pp. 60--68, Stanford, 1998.
- [JIN99] H. Jing, K. McKeown. The Decomposition of Human-Written Summary Sentences. *In Proceedings of the 22<sup>th</sup> ACM SIGIR Conference on Research and Development in Information Retrieval*. 1999.

- [JON96] K. S. Jones, J. R. Galliers. Evaluating Natural Language Processing systems: an Analysis and review. *Edition Springer*, New York, 1996.
- [KAL88] G. Källgren. Automatic Abstracting on Content in text. *Nordic Journal of Linguistics*. pp. 89-110, Vol. 11, 1988.
- [KAN98] M. Y. Kan, J. Klavans. Linear Segmentation and segment Significance. In *Proc. 6th Workshop on Very Large Corpora (WVLC-98)*, pp. 197--205, Montreal, Canada, August. ACL SIGDAT.
- [KLA95] J. L. Klavans, J. Shaw. Lexical Semantics in summarization. In *Proceedings of the First Annual Workshop of the IFIP Working Group for NLP and KR*, Nantes, France, 1995.
- [KLA98] J. Klavans, K. R. McKeown, M. Y. Kan. Ressources for Evaluation of Summarization Techniques. In *acts of First International Conference on Language Ressources & Evaluation (LREC)*, Grenade, Espagne, pp. 899--902, 1998.
- [KOZ93] H. Kozima. Text Segmentation Based on Similarity between Words, in *Proceedings of the ACL*, pp. 286--288, 1993.
- [KUP95] J. Kupiec, J. Pedersen, F. Chen. A Trainable Document Summarizer. In *Proceedings of the 18<sup>th</sup> ACM SIGIR Conference on Research and Development in Information Retrieval*. pp. 68--73, 1995.
- [LIT95] D. J. Litman, R. J. Passoneau. Combining Multiple Knowledge Sources for discourse segmentation. *Actes of the 33th proceedings on ACL*, pp.108--115, 1995.
- [LUH58] P. H. Luhn. Automatic creation of literature abstracts. *IBM Journal*. pp. 159--165, 1958.
- [MAN97] I. Mani, E. Bloedern. Multi-Document Summarization by Graph Search and Merging. In *proceedings of AAAI-97*, pp.622--628, 1997.
- [MaT87] W. C. Mann, S. A. Thompson. Rhetorical Structure Theory: A Theory of Text Organization. *Technical Report, isi Reprint Series isi/RS-87-190*, Information Sciences Institute, Marina Del Rey, 1987.
- [MAR97a] D. Marcu. The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts. *Ph.D. thesis*, Department of Compute Science, University of Toronto, Toronto, Canada, 1997.
- [MAR97b] D. Marcu. From Discourse Structures to Text Summaries. In *Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*. pp. 82--88, Espagne, 1997.

- [MAR99a] D. Marcu, Discourse Trees are good indicators of importance in text. *In Inderjeet Mani and Mark Maybury, editors, Advances in Automatic Text Summarization*. The MIT Press.
- [MAR99b] D. Marcu. The Automatic Construction of Large-Scale Corpora for Summarization Research. *In Proceedings of the 22<sup>th</sup> ACM SIGIR Conference on Research and Development in Information Retrieval*. pp. 137--144, 1999.
- [McK95] K. McKeown, J. Robin, K. Kukich. Designing and Evaluating a new Revision-Based Model for Summary Generation. *Information Proceedings and Management*, Vol. 3, N° 5, 1995.
- [MIT97] M. Mitra, A. Singhal, C. Buckley. Automatic Text Summarization by Paragraph Extraction. *In Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*. pp. 31--36, Espagne, 1997.
- [MiK99] V. O. Mittal, M. Kantrowitz, J. Goldstein, J. Carbonnel. Selecting Text Spans for Document Summaries: Heuristics and Metrics. *In Proceedings of AAAI-99*, Orlando, Juillet 1999.
- [MITT94] E. Mittendorf, P. Schäuble. Document and Passage Retrieval Based on Hidden Markov Models. *In Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. pp. 318--327, Dublin, Ireland, Juillet 1994.
- [MOF94] A. Moffat, R. Sacks-Davis, R. Wilkinson, J. Zobel. Retrieval of Partial Documents. *In Proceedings of the Second Text Retrieval Conference (TREC-2)*, 1994.
- [MOR91] J. Morris, G. Hirst. Lexical Cohesion Computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, pp. 21--48, Vol. 17, N° 1, 1991.
- [NAK95] C. H. Nakatani, B. Grosz, D. Ahn, J. Hirschberg. Instructions for annotating discourse. *Technical Report 21-95, Center for Research in Computing Technology*, Harvard University, Cambridge, MA, 1995.
- [NIST93] NIST, "TIPSTER Information-Retrieval Text Research Collection, on CD-ROM", *published by the National Institute of Standards and Technology*, Gaihersburg, Maryland, 1993.
- [OCO80] J. O'Conoor. Answer-passage Retrieval by Text Searching. *Journal of the American Society for Information Science*. Vol. 31, N° 4, pp.227--239, 1980.
- [PAI90] C. D. Paice. Constructing Literature Abstracts by Computer: Techniques and Prospects. *Info. Proc. And Management*. Vol. 26, pp. 171--186, 1990.

- [PON97] J.M. Ponte, W.B. Croft. Text Segmentation by Topic. *In Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*. pp. 120--129, 1997.
- [RAD98] D. Radev, K. McKeown. Generating Natural Language Summaries from Multiple Online Sources. *Computational Linguistics*, 1998.
- [RAT61] G. J. Rath, A. Resnick, T.P. Savage. The Formation of Abstracts by the Selection of Sentences. *American Documentation*. Vol. 12, N° 2, pp. 139--143, Avril 1961.
- [RO88] J. S. RO, An Evaluation of the Applicability of Ranking Algorithms to Improve the Effectiveness of Full-Text Retrieval. *Journal of the American Society for Information Science*. Vol. 39, N° 2, pp. 73--78, 1988.
- [SAL91] G. J. Salton, C. Buckley. Automatic text Structuring and Retrieval – Experiments in Automatic Encyclopedia Searching. *In Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*. pp. 21--30, Chicago, 1991.
- [SAL93a] G. J. Salton, J. Allan, C. Buckley. Approaches to passage Retrieval in Full Text Information Systems. *In Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. pp. 49--58, 1993.
- [SAL94a] G. J. Salton, A. Singhal. Automatic Text Theme Generation and Analysis of Text Structure. *Cornell Computer Science Technical Report 94-1438*. Juillet 1994.
- [SAL94b] G. J. Salton, C. Buckley, A. Singhal. Automatic Analysis. Theme Generation and Summarization of Machine Readable Texts. *Science*. Vol. 264, pp. 1421--1426. 1994.
- [SAL94c] G. J. Salton, C. Buckley. Automatic Structuring and Retrieval of Large Text Files. *Communications of ACM*. Vol. 37, N° 2, pp. 97--108. 1994.
- [SAL94d] G. J. Salton, J. Allan. Automatic Text Decomposition and Structuring. *Actes of RIAO'94*, pp. 6--29, New-York, 1994.
- [SAL96] G. J. Salton, A. Singhal, C. Buckley, M. Mitra. Automatic Text Decomposition Using Text Segments and Text Themes. *Proceedings of the Seventh ACM Conference on Hypertext*. Washington D.C., 1996.
- [SKO72] E. F. Skorokhod'ko. Adaptive Method of Automatic Abstracting and Indexing. *In IFIP Congress*. pp. 1179--1182, Hollande, 1972.
- [SpJ93] K. Spark Jones. Discourse Modeling for Automatic Summarizing. *Technical Report 29D*, Computer Laboratory, University of Cambridge, 1993.

- [STA92] C. Stanfill, D. L. Waltz. Statistical Methods, Artificial Intelligence and Information Retrieval. *Text-based intelligent systems*. pp. 215--225, 1992.
- [STR98] T. Strzalkowski, J. Wang, B. Wise. A Robust Practical Text Summarization System. *In AAAI Intelligent Summarization Workshop*. pp. 26--30, 1998.
- [SUM98] "TIPSTER Text Summarization Evaluation Conference (SUMMAC)", [http://www-nlpir.nist.gov/related\\_projects/tipster\\_summac/](http://www-nlpir.nist.gov/related_projects/tipster_summac/)
- [TAI83] J. I. Tait. Automatic Summarizing of English Texts. *Ph.D. thesis*, University of Cambridge UK, 1983.
- [TEU97] S. Teufel, M. Moens. Sentence Extraction as a Classification Task. *In Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*. pp. 58--65, Espagne, 1997.
- [TEU98] S. Teufel, M. Moens. Sentence Extraction and Rhetorical Classification for Flexible Abstracts. *Spring Symposium on Intelligent Text summarization*, Stanford, 1998.
- [TIP98] Tipster Text Phase III workshop, 1998.
- [XU96] J. Xu, W.B. Croft. Query Expansion Using Local and Global Document Analysis. *In Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. pp. 4--11, Zurich, Swiss, 1996.
- [VaM98] P. Van Mulbregt, I. Carp, L. Gillick, S. Lowe, J.P. Yamron. Text Segmentation and Topic Tracking on Broadcast News via a Hidden Markov Model Approach. *In Proceedings of the ICSLP*. Vol. 6, pp. 2519--2522, 1998.
- [WAL91] W. Walker. Redundancy in collaborative dialogue. *Actes of AAAI Symposium on Discourse Structure in Natural Language Understanding and Generation*. Pacific Grove, USA, 1991.
- [WIL94] R. Wilkinson. Effective Retrieval of Structured Documents. *In Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 311--317, 1994.
- [WU94] D. Wu, Aligning a parallel english-chinese corpus statistically with lexical criteria. *In proceedings of ACL-94*, 1994.
- [YAM98] J.P. Yamron, I. Carp, L. Gillick, S. Lowe, P. Van Mulbregt. A Hidden Markov Model Approach to Text Segmentation and Event Tracking. *IEEE ICASSP*. 1998.
- [ZEC95] K. Zechner. Automatic Text Abstracting by Selecting Relevant Passages. *MSc Dissertation in Cognitive Science and Natural Language*.





## **Partie III**

### **Nouveaux Modèles**

# Chapitre 6

## Modèles de séquences pour l'extraction d'information avec des requêtes fermées

**Résumé.** Dans ce chapitre nous considérons la modélisation des séquences de termes par des Perceptrons Multi-Couches (PMC) et des modèles de markov cachés (MMC) et nous montrons comment ces modèles peuvent être utilisés pour s'acquitter de tâches d'Extraction d'Information de Surface et de Surlignage (tâches qui n'ont pas besoin d'analyses sémantique et syntaxique complexes). Nous étudions différentes représentations de textes et analysons différentes contraintes grammaticales sur les sorties des modèles. Nous présentons un ensemble de résultats obtenus sur la base MUC-6. Afin de valider notre étude, nous présentons une méthode d'estimation statistique pour obtenir des intervalles de confiances sur nos résultats.

**Mots clés :** Réseaux de Neurones, Modèles de markov cachés, Extraction d'Information, séquences de termes, validation.

### 6.1 Introduction

Avec le développement de systèmes de communication électronique de plus en plus performants, nous assistons à l'émergence de nouvelles demandes pour des systèmes de fouille de données. Beaucoup de nouvelles tâches se situent entre la Recherche d'Information (RI) et l'Extraction d'Information (EI). Les systèmes d'apprentissage dynamique jouent un rôle central dans le développement de ces champs mais jusqu'à présent ils ont été surtout utilisés pour l'amélioration des systèmes existants.

Nous proposons d'étendre les capacités des modèles statistiques de RI pour traiter des tâches plus complexes en recherche et en extraction d'information. Pour cela, nous allons explorer l'utilisation de modèles de séquences probabilistes adaptés à l'analyse de séquences textuelles. Nous allons considérer en particulier le texte comme une séquence de mots et non comme un ensemble non ordonné de termes (sac de mots). Dans cette perspective, nous allons proposer deux modèles de séquences qui permettent de travailler à un niveau plus fin de ce que font les modèles classiques de recherche.

Comme nous le montrerons, ces modèles sont capables de traiter différentes sous-tâches de la fouille de données.

Le chapitre est organisé de la façon suivante. Dans un premier temps, nous présentons dans le paragraphe 6.2 le formalisme des modèles de séquences employés ainsi que le codage des termes. Pour représenter les séquences de mots, nous choisirons de représenter les termes dans un espace de petite dimension, cette représentation s'est avérée efficace pour des tâches complexes [ZAR99]. Nous allons ensuite introduire au paragraphe 6.3 des modèles de séquences probabilistes qui permettent de prendre en compte différentes tâches avec des requêtes fermées. Par tâches avec requêtes fermées nous entendons les tâches où l'information utile est connue à l'avance de façon à ce qu'elle puisse être utilisée pour construire le système, à l'opposé par exemple de la tâche ad-hoc en RI. Nous avons implémenté des modèles de ce type avec des réseaux de neurones et des Modèles de Markov Cachés. Dans le paragraphe 6.4, nous présenterons en détail les systèmes utilisés et nos expériences. Nous décrirons tout d'abord le corpus utilisé ; il s'agit d'articles du Wall Street Journal de MUC-6 [MUC6] et leurs patrons associés. Puis nous préciserons les tâches visées, le surlignage et l'extraction d'information de surface. La première tâche consiste à sélectionner les séquences de mots les plus pertinentes dans un document. La seconde consiste à étiqueter les mots en regard de plusieurs centres d'intérêts. Nous nous intéressons, pour le surlignage, à détecter les descriptions de changements de personnes aux seins des entreprises (promotions, démissions ou fin de carrière) et, dans le cas de l'extraction, à étiqueter le *nom* et la *position* de la personne concernée. Nous discuterons ensuite l'introduction de contraintes grammaticales. Enfin, nous décrirons en détail nos résultats et montrerons le comportement de nos systèmes en les comparant avec des modèles plus classiques.

## 6.2 Analyse de séquences

### 6.2.1 Formalisme

Les modèles de séquences auxquels nous nous intéressons ici traduisent une séquence de symboles en entrée en une séquence de sortie. Notons respectivement  $x_{1,n}=x_1x_2 \dots x_n$ ,  $x_i \in \mathbf{X}$  et  $t_{1,n}=t_1t_2 \dots t_n$ ,  $t_i \in \mathbf{T}$  les séquences d'entrée et de sortie, où  $\mathbf{X}$  et  $\mathbf{T}$  sont les espaces d'entrée et de sortie,  $n$  la longueur de ces séquences et  $x_i$  et  $t_i$  indiquent respectivement le  $i^{\text{ème}}$  élément de la séquence d'entrée et de sortie. Notre but est de trouver la séquence de sortie la plus probable  $t_{1,n}$  c'est-à-dire celle qui maximise la probabilité  $p(t_{1,n} / x_{1,n})$ . Dans un souci de simplification, nous écrirons  $p(t_i / x_j)$  au lieu de  $p(t_i = T_k / x_j = X_p)$ , où  $X_p$  et  $T_k$  indiquent respectivement un élément d'entrée et de sortie.

On peut ramener les problèmes de surlignage et d'extraction à un problème d'analyse de séquences, si l'on suppose qu'il existe une correspondance unilatérale entre les mots et les *symboles de sortie* ainsi qu'entre les symboles de sortie et les *classes*. Dans ce cas, la séquence d'entrée est le texte (ou une représentation du texte) et la séquence de sortie correspond aux étiquettes qui codent l'alphabet des classes contenues dans le texte. Dans ce cadre, les mots sont représentés par des vecteurs ( $x_i$  est alors la représentation vectorielle du  $i^{\text{ème}}$  élément de la séquence) et  $t_i$  l'étiquette de sa classe.  $\mathbf{X}$  est l'espace vectoriel des termes et  $\mathbf{T}$  celui de l'ensemble des classes.

## 6.2.2 Caractérisation de séquences

Un problème crucial est posé par la dimension de l'espace de représentation des termes. En effet, avec des modèles de séquences, nous cherchons à modéliser la dépendance des termes dans la séquence et donc à estimer des densités de probabilités. Il est donc souhaitable de représenter ces termes dans un espace de représentation de faible dimension. Notre approche est assez similaire à celle proposée précédemment par [MITT94] pour des tâches avec requêtes ouvertes. Elle se situe à l'opposé de l'approche de [BIK99, FRE99] qui considèrent les termes successifs d'une séquence indépendants. Comparée cette dernière, notre approche offre plusieurs avantages :

1. Une estimation plus robuste (grâce à la réduction de la dimension de l'espace de représentation),
2. La possibilité d'utiliser des modèles continus,
3. La possibilité de considérer des informations additionnelles en entrée notamment le contexte local des termes.

Pour la représentation des termes nous allons tirer parti de l'étiquetage des documents. Nous présentons tout d'abord la mesure continue  $U$  [AND92] qui s'est avérée efficace dans nos expériences. Ensuite, nous introduirons une information supplémentaire, l'information morpho-syntaxique. Enfin, nous évaluerons l'apport des différentes caractéristiques envisagées avec des techniques de sélection de variables.

### 6.2.2.1 Le codage continu

Nous considérons la phrase comme l'unité de base du texte. Un texte est représenté comme une séquence de termes, dont chaque terme est codé par un vecteur. Pour chaque mot  $m_i$  du corpus, désignons par  $p$  et  $p'$  les nombres respectifs de phrases pertinentes et non-pertinentes dans lesquelles le mot apparaît. Notons également par  $q$  et  $q'$  les nombres respectifs de phrases pertinentes et non-pertinentes dans lesquelles le mot ne figure pas. La mesure  $U$  du mot  $m_i$  est alors définie par le rapport :

$$U(m_i) = \sqrt{N} \cdot \frac{pq' - p'q}{\sqrt{(p+p')(p+q)(p'+q')(q+q')}}$$

où  $N$  désigne le nombre total de phrases ( $N = p+p'+q+q'$ ). La mesure  $U$  d'un terme est basée sur ses fréquences d'occurrence dans les phrases pertinentes et non-pertinentes, et pour cette raison elle traduit, dans une certaine mesure, la *saillance* d'un terme par rapport à une tâche définie.

### 6.2.2.2 Information Morpho-syntaxique

Afin d'introduire une représentation plus riche, nous avons également utilisé les étiquettes morpho-syntaxiques des termes [AMI99a]. L'idée est que les étiquettes syntaxiques doivent permettre de distinguer les termes qui ont mêmes valeurs  $U$ , ce qui doit contribuer à une meilleure représentation des mots. De ce fait, nous avons augmenté l'espace des représentations par des étiquettes morpho-syntaxiques. Nous avons utilisé pour cela un étiqueteur syntaxique en libre disponibilité sur Internet

[SCH94]. Dans un premier temps nous avons considéré 7 étiquettes syntaxiques : *Nom* (*N*), *Nom Propre* (*NP*), *Verbe* (*V*), *Adjectif* (*Adj*), *Complément* (*CC*), *Déterminant* (*Dét*) et *Autres* (*A*) pour les autres étiquettes. Chaque terme est ainsi représenté par un vecteur de dimension 8, dont la première composante est la mesure *U* du terme, et les 7 autres sont mises à 0 sauf celle correspondante à l'étiquette syntaxique du terme, qui est fixée à 1.

### 6.2.2.3 Sélection de Variable

Les composantes n'apportent pas toutes autant d'information. Afin d'évaluer leur pertinence nous avons utilisé une méthode de sélection de variables automatique sur l'ensemble des caractéristiques [AMI99b]. Nous avons choisi une méthode proposée par [BON96] pour éliminer toutes les caractéristiques non-informatives. La pertinence d'un ensemble de variables d'entrée est définie par l'information mutuelle entre ces variables et leur classe correspondante. Cette mesure de dépendance est bien adaptée pour calculer les dépendances non linéaires comme celles capturées par des Réseaux de Neurones. Pour deux variables *x* et *t*, l'information mutuelle est définie comme :

$$MI(x,t) = \sum_{x,t} p(x,t) \cdot \log \frac{p(x,t)}{p(x)p(t)}.$$

Dans notre cas, *x* dénote le codage du mot et *d* indique la classe associée à *x* (e.g. Non-Pertinent, Personne, Position pour la tâche d'extraction et Pertinent, Non-Pertinent pour le surlignage). En commençant par un ensemble vide, les variables sont ajoutées au fur et à mesure, la variable *x<sub>i</sub>* choisie à l'étape *i* étant celle qui maximise  $MI(Sv_{i-1}, x_i, t)$  où  $Sv_{i-1}$  est l'ensemble des *i-1* variables déjà sélectionnées. La sélection s'arrête lorsque le rapport  $\rho = MI(Sv_{i-1}, t) / MI(Sv_i, t)$  passe au-dessus d'un seuil fixé (0.99 dans notre cas). Nous avons expérimenté cette méthode de sélection de variables sur les deux tâches de surlignage et d'extraction. Dans les deux cas, nous avons obtenu le même ensemble de 5 variables *U*, *NP*, *N*, *ADJ*, *V*. Le tableau 2 donne pour chaque étape les variables sélectionnées,  $MI(Sv_i, t)$  et  $\rho$ , pour le surlignage. Dans la suite nous utiliserons les deux représentations *U* et (*U*, *NP*, *N*, *ADJ*, *V*).

Etape <i>i</i>	Variable <i>x<sub>i</sub></i>	$MI(Sv_i, d)$	$\rho$
1	<i>U</i>	0.1779	
2	<i>NP</i>	0.1871	0.9506
3	<i>N</i>	0.1943	0.9633
4	<i>ADJ</i>	0.2009	0.9667
5	<i>V</i>	0.2067	0.9719
6	<i>DET</i>	0.2084	0.992
7	<i>A</i>	0.2092	0.9962
8	<i>CC</i>	0.2092	1

Tableau 2. La sélection de variables pour le surlignage.

### 6.3 Modèles de Séquences

Notre but est d'assigner des étiquettes de classe aux mots du texte. Ce problème peut être formellement défini comme la recherche de la meilleure séquence d'étiquettes  $t_{1,n}$ . Cela revient à déterminer la séquence d'étiquettes la plus probable, associée à une séquence de mots :

$$\hat{t}_{1,n} = \arg \max_{t_{1,n}} p(t_{1,n}/x_{1,n}) \quad (6.1)$$

L'équation (6.1) peut se mettre sous une forme plus commode:

$$\hat{t}_{1,n} = \arg \max_{t_{1,n}} \frac{p(t_{1,n}, x_{1,n})}{p(x_{1,n})} \quad (6.2)$$

$$= \arg \max_{t_{1,n}} p(t_{1,n}, x_{1,n}) \quad (6.3)$$

Comme  $p(x_{1,n})$  est constante pour tous les  $t_{1,n}$ , les équations (6.2) et (6.3) sont équivalentes. La probabilité jointe dans l'équation (6.3) peut être classiquement décomposée en un produit de probabilités conditionnelles :

$$p(t_{1,n}, x_{1,n}) = p(x_1) \cdot p(t_1/x_1) \cdot p(x_2/t_1, x_1) \cdot p(t_2/t_1, x_{1,2}) \quad (6.4)$$

$$\dots p(x_n/t_{1,n}, x_{1,n-1}) \cdot p(t_n/t_{1,n-1}, x_{1,n-1})$$

$$= p(x_1) p(t_1/x_1) \cdot \prod_{i=2}^n p(x_i/t_{1,i-1}, x_{1,i-1}) \cdot p(t_i/t_{1,i-1}, x_{1,i}) \quad (6.5)$$

$$= \prod_{i=1}^n p(x_i/t_{1,i-1}, x_{1,i-1}) \cdot p(t_i/t_{1,i-1}, x_{1,i}) \quad (6.6)$$

Nous avons simplifié l'équation (6.5) pour obtenir l'équation (6.6) en définissant des termes comme  $t_{1,0}$ , ainsi que leur probabilité. L'équation (6.6) est déduite de (6.4) par le développement de  $p(t_{1,n}/x_{1,n})$  par  $p(x_1)$ . De façon symétrique nous pouvons aussi développer (6.4) en commençant par  $p(t_1)$  :

$$p(t_{1,n}, x_{1,n}) = \prod_{i=1}^n p(t_i/t_{1,i-1}, x_{1,i-1}) \cdot p(x_i/t_{1,i}, x_{1,i-1}) \quad (6.7)$$

En suivant [CHA 93], nous allons dériver des équations (6.6) et (6.7) deux expressions correspondant à deux familles de modèles dynamiques. Nous allons nous intéresser à l'équation (6.6), en considérant tout d'abord les deux hypothèses suivantes, de localité et d'indépendance :

$$p(x_i/t_{1,i-1}, x_{1,i-1}) = p(x_i/x_{1,i-1}) \quad (6.8)$$

$$p(t_i/t_{1,i-1}, x_{1,i}) = p(t_i/x_{1-k, i+k}) \quad (6.9)$$

avec ces hypothèses, l'équation (6.6) s'écrit :

$$\hat{t}_{1,n} = \arg \max_{t_{1,n}} \prod_{i=1}^n p(x_i / x_{1,i-1}) \cdot p(t_i / x_{i-k,i+k}) \quad (6.10)$$

$$= \arg \max_{t_{1,n}} \prod_{i=1}^n p(t_i / x_{i-k,i+k}) \quad (6.11)$$

L'interprétation de l'équation (6.11) est simple : la séquence optimale est obtenue en choisissant l'état le plus probable pour chaque mot.

En partant de l'équation (6.7), et en utilisant maintenant les hypothèses :

$$p(t_i / x_{1,i-1}, t_{1,i-1}) = p(t_i / t_{i-1}) \quad (6.12)$$

$$p(x_i / x_{1,i-1}, t_{1,i}) = p(x_i / t_i) \quad (6.13)$$

On obtient :

$$\hat{t}_{1,n} = \arg \max_{t_{1,n}} \prod_{i=1..n} p(x_i / t_i) \cdot p(t_i / t_{i-1}) \quad (6.14)$$

Les expressions (6.11) et (6.14) correspondent à deux modèles différents pour le décodage de la meilleure séquence des étiquettes. Les hypothèses sur lesquelles ils reposent se justifient en grande partie par la transformation du problème de décodage combinatoire initial en un problème plus simple. Outre l'implémentation de (6.11), nous avons aussi mené des expériences avec le modèle suivant :

$$\hat{t}_{1,n} = \arg \max_{t_{1,n}} \prod_{i=1..n} p(t_i / x_{i-k,i+k}) \cdot p(t_i / t_{i-1}) \quad (6.15)$$

qui est dérivé de (6.11) par ajout de probabilités de transition  $p(t_i / t_{i-1})$ . Dans ce cas, nous supposons qu'il existe une grammaire sur les séquences d'étiquettes. Cette modélisation permet de prendre en compte des informations de transition en introduisant des contraintes grammaticales.

Les PMC et les MMC permettent d'implémenter les équations (6.15) et (6.14). Les PMC peuvent estimer aussi bien des distributions de probabilités continues que discrètes. De plus ces modèles permettent facilement d'introduire une information contextuelle. Dans (6.11) et (6.15) la classe d'un mot dépend en effet d'un contexte de  $k$  mots à droite et à gauche du mot courant, ainsi que du mot lui-même. Ceci peut être facilement mis en œuvre en définissant l'entrée du réseau comme une fenêtre de mots  $x'_i = (x_{i-k}, \dots, x_i, \dots, x_{i+k})$ . Par ailleurs, les MMC sont par nature des modèles de séquence permettant de modéliser des processus temporels tels que des séquences de termes dans notre cas. On peut les utiliser également pour estimer des distributions de probabilités continues ou discrètes.

Afin d'évaluer ces deux modèles de séquences, nous avons implémenté un ensemble de modèles MMC discrets similaires à ceux proposés dans [FRE99]. Ces modèles opèrent dans un espace discret, ils ne prennent pas en entrée une représentation continue (comme celle proposée en 6.2.2.1), mais une séquence de termes  $w$  correspondants à un simple pré-traitement de la séquence de mots du document (en éliminant les contredictionnaires et en normalisant). Ce modèle requiert l'estimation de  $p(x_i / C_k = t_i)$  pour

chaque terme  $x_i$  et chaque classe  $C_k$ . Nous avons utilisé le modèle de Naïve Bayes lissé de la section (3.4.1.1) :

$$\hat{p}(x_i / C_k) = \frac{tf(x_i / C_k) + 1}{N + \sum_{j=1}^c tf(x_i / C_j)} \quad (6.16)$$

où  $tf(x_i/C_j)$  représente le nombre d'occurrences du terme  $x_i$  dans la classe  $C_j$ , et  $c$  le nombre total de classes. Ce modèle a été proposé par [VAP82] avec l'hypothèse que l'observation de chaque terme est *a priori* égale.

## 6.4 Expériences

### 6.4.1 Corpus

Nous avons utilisé la base étiquetée MUC-6 [MUC6] pour nos expériences. Ce corpus est constitué d'une collection de documents de 200 articles du Wall Street Journal analysés par des experts humains. Pour chaque document, un ensemble de patrons décrit les instances de changements de personnel (promotion, démission, etc) (Figure 36). Les patrons sont divisés en champs décrivant un aspect particulier de l'événement (Figure 36, bas gauche). Dans le travail présenté ici, nous étudions seulement deux champs : le *Nom* et la *Position* de la personne concernée.

Nous avons mis au point une procédure pour étiqueter automatiquement chaque terme avec une étiquette de classe. Comme chaque document a un ensemble de patron associé, et des marqueurs de fins paragraphes, nous comparons chaque paragraphe à chaque patron de document. Si un paragraphe s'apparie partiellement à au moins deux champs du patron, ce paragraphe est considéré comme pertinent. Les paragraphes sont alors divisés en groupe de mots (une séquence d'au moins deux mots séparés par une ponctuation ou des conjonctions). Si un groupe de mots contient le mot de champ du patron, à tous les mots de ce groupe sont donnés la classe du champ. Tous les autres mots sont étiquetés comme non pertinents. Cette heuristique implique quelques erreurs d'étiquetages, comme les insertions et les remplacements. Nous considérons ce biais comme du bruit sur les données. Notre but est de mettre en place des modèles capables de gérer les erreurs d'étiquetage correspondant aux systèmes d'étiquetage automatique.

Cette approche d'étiquetage des corpus peut être adaptée à d'autres situations. Il existe beaucoup de cas où l'on a accès à une base de donnée contenant des champs textuels et un corpus de documents pour lequel l'information peut être extraite.



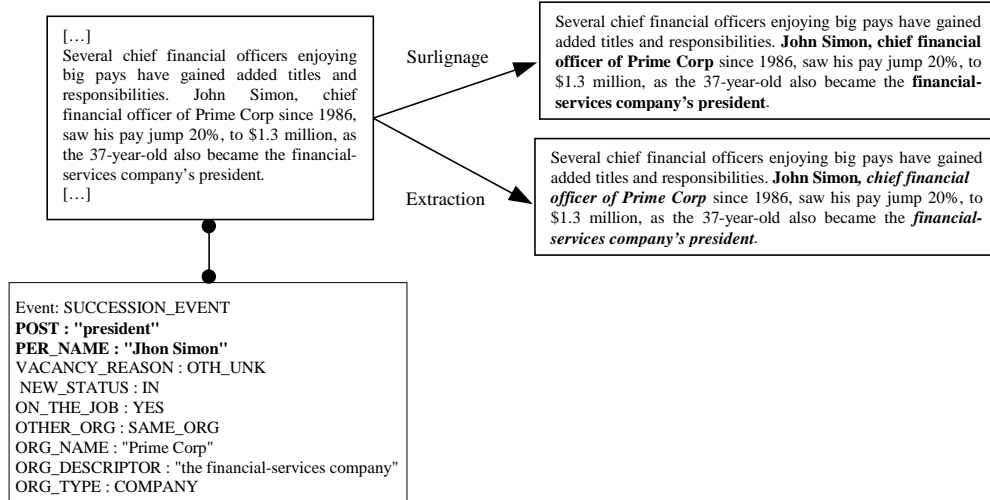


Figure 36. Un paragraphe (haut gauche) est comparé avec un patron (gauche bas) pour obtenir les étiquettes des mots (haut droite). Dans le patron nous avons indiqué en gras les deux champs que nous utilisons : POST et PER\_NAME. Pour la tâche d'extraction les mots d'étiquettes PERSONNE sont montrés en gras et les mots d'étiquettes POSITION sont montrés en italiques. Pour la tâche de surlignage les termes d'étiquettes PERSONNE et POSITION sont montrés en gras. Les mots non-pertinents sont montrés en normal.

Dans les expériences décrites sur le surlignage et l'extraction, le corpus est divisé en deux bases, d'apprentissage et de test, correspondant respectivement à 100 et 105 paragraphes. A peu près 8% des mots sont de classe « Personne », 32% de classe « Position » et 60% sont non-pertinents.

#### 6.4.2 Extraction et Surlignage

Les deux exemples d'applications auxquelles nous nous intéressons sont illustrés par la Figure 36 (en haut à droite). Ils sont :

- **le surlignage.** Il s'agit ici de détecter toutes les descriptions d'événements de changement de positions, c'est-à-dire d'étiqueter des sous-séquences de textes comme non-pertinent (*Irrelevant*) et pertinent (*Relevant*) (les étiquettes des termes prennent deux valeurs 0/1),
- **l'extraction.** Ici nous nous proposons d'extraire le nom et la position de la personne concernée, c'est-à-dire d'étiqueter les sous-séquences comme Personne (*Per*), Position (*Pos*) et non-Pertinent (*I*) (les étiquettes des termes pourront prendre 3 valeurs).

Pour le surlignage et l'extraction, nous avons utilisé une grammaire pour contraindre les séquences autorisées, nous discutons de ce point dans le paragraphe 6.4.3. Comme entrée à ces modèles nous avons utilisé les deux représentations que nous avons déjà présentées, la mesure *U* seule ou avec l'étiquetage morpho-syntaxique. Nous

montrons que l'information morpho-syntaxique permet d'améliorer sensiblement les performances [AMI99a].

Nous supposons que toutes les occurrences des mots tombent exactement dans une classe, dans le cas de surlignage dans une des deux classes *pertinent* et *non-pertinent* et dans le cas de l'extraction dans une des trois classes *personne*, *position* et *non-pertinent*. Outre les deux modèles déjà présentés, nous avons également implémenté des MMC discrets (6.16) décrit en section 6.3. Pour les trois modèles, la segmentation de textes en différentes classes a été réalisée grâce à l'algorithme de Viterbi.

### 6.4.3 Grammaires

Pour les deux tâches étudiées, les sorties du modèle peuvent être contraintes, ou pas, à produire certains types de séquences. Les contraintes correspondent à une connaissance *a priori* sur la tâche ou sur le type d'information que l'utilisateur cherche. Elles peuvent être décrites comme des grammaires sur les sorties du modèle. Pour nos expériences, nous avons testé différentes contraintes [AMI99b, GAL00].

Pour le surlignage, nous avons utilisé l'approche proposée par [MITT94] qui consiste à prendre une grammaire de type *I-R-I*, i.e. le modèle est forcé à étiqueter le texte comme la succession d'un passage non-pertinent (*I*) suivi d'un passage pertinent (*R*) et se terminant par un passage non-pertinent. Si ce schéma ne correspond pas tout à fait à l'information pertinente présente dans le corpus, il fournit cependant un seul résumé de texte qui peut être intéressant. Il peut être vu aussi comme une première approximation à notre paradigme de surlignage.

Nous avons aussi utilisé une grammaire de type  $\langle I / R^{(3)} \rangle$  qui étiquette les mots alternativement comme pertinent ou non-pertinent, ( $\langle \rangle$  indique une ou plusieurs occurrences et  $|$  dénote le *ou* logique), les séquences pertinentes ayant une durée minimale de 3 termes. Le nombre 3 a été choisi de façon heuristique.

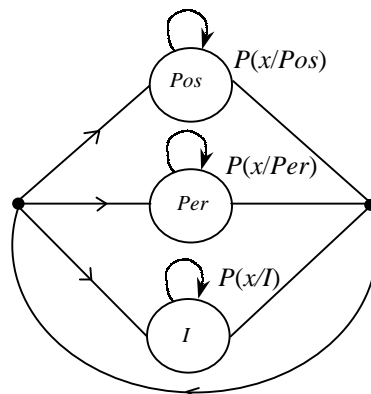


Figure 37. Un modèle MMC correspondant à la topologie  $\langle I/Per^{(3)}|Pos^{(3)} \rangle$  pour la tâche d'extraction (le modèle de durée n'est pas montré). Les distributions de probabilité associées aux états correspondent aux différentes classes, Personne (Per), Position (Pos) et non-pertinent (I).

De la même manière, dans le cas de l'extraction, nous avons utilisé des grammaires de type  $I-\langle Per-Pos-I \rangle$ , et  $\langle I|Per^{(3)}|Pos^{(3)} \rangle$  correspondant à une configuration très simple d'un MMC à trois états montré dans la figure 37. Le premier et le deuxième états montrent la distribution de probabilité des termes en regard des classes *Personne* (*Per*) et *Position* (*Pos*), le troisième état représente le choix d'un mot du « dictionnaire général » et il est donc non-pertinent (*I*) pour notre tâche d'extraction.

## 6.4.4 Evaluation

### 6.4.4.1 Comparaison des représentations $U$ et ( $U, NP, N, ADJ, V$ )

Le tableau 3 résume les résultats d'expériences en surlignage et extraction. Il donne le pourcentage de bonne classification des deux modèles (6.14) et (6.15) notés respectivement  $U_{PMC}$  et  $U_{MMC}$ , avec deux choix de représentation des termes,  $U$  et  $U$  plus les étiquettes morfo-syntaxiques ( $U+tag$ ).

Dans ces expériences nous avons utilisé les grammaires  $I-R-I$  et  $I-\langle Per-Pos-I \rangle$  pour le surlignage et l'extraction. Les systèmes  $U_{PMC}$  et de  $U_{MMC}$  montrent des comportements distincts, bien qu'il ne soit pas clair à partir de ces expériences de déterminer lequel est le meilleur. Les performances moyennes sont néanmoins supérieures avec le  $U_{PMC}$ .

Surlignage		% bonne classification		Extraction		% bonne classification		
		<i>Pertinent</i>	<i>Moyenne</i>			<i>Position</i>	<i>Personne</i>	<i>Moyenne</i>
$U_{MMC}$	$U + I-R-I$	65.59	64.09	$U + I-\langle Per-Pos-I \rangle$	55.86	18.27	57	
	$U + tag + I-R-I$	<b>85.15</b>	62.73	$U + tag + I-\langle Per-Pos-I \rangle$	<b>87.49</b>	12.29	57.37	
$U_{PMC}$	$U + I-R-I$	79.71	81.10	$U + I-\langle Per-Pos-I \rangle$	54.75	<b>51.27</b>	67.54	
	$U + tag + I-R-I$	81.06	<b>82.45</b>	$U + tag + I-\langle Per-Pos-I \rangle$	61.58	38.34	<b>70.20</b>	

Tableau 3. Les performances de  $U_{MMC}$  et de  $U_{PMC}$  pour le surlignage et l'extraction. Dans le premier cas, la moyenne est par rapport aux classes pertinent et non-pertinent. Tandis que dans le second elle est par rapport aux classes *Position*, *Personne* et non-pertinent. Dans chaque colonne les meilleures performances sont en gras.

Pour le surlignage, l'utilisation d'informations syntaxiques améliore sensiblement les performances de la classe pertinente tandis que la performance moyenne décroît légèrement. Pour la tâche d'extraction, l'information syntaxique améliore clairement l'étiquetage de la classe *Position*, tandis que la classe *Personne* n'est bien reconnue par aucune des deux représentations.

La figure 38 montre les courbes Précision-Rappel obtenues avec le  $U_{PMC}$  sans contrainte sur les sorties, où les termes sont représentés avec la mesure  $U$  (à gauche) ou par leur mesure  $U$  plus leur étiquette syntaxique (à droite). Pour chaque représentation, nous avons donné les courbes pour la classe « *Personne* », « *Position* » (tâche d'extraction) ainsi que pour la classe pertinente (la tâche de surlignage). La précision est définie comme  $c_k/M$  et le rappel comme  $c_k/N_k$  où  $c_k$  est le nombre de termes bien classés de la

classe  $k$ ,  $M$  le nombre total de termes jugés pertinent par le système et  $N_k$  le cardinal de la classe  $k$ . Tous les mots sont triés dans l'ordre de leur pertinence, étiquetant les  $M$  plus élevés comme pertinent et le reste comme non-pertinent. Nous constatons que l'utilisation d'informations syntaxiques améliore sensiblement les performances et qu'elles ont plus d'effet pour des rappels bas (précisions élevées).

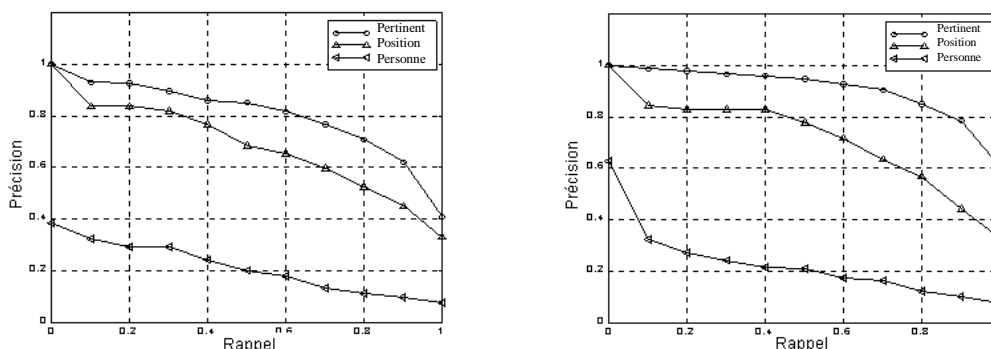


Figure 38. Courbes de Précision-Rappel obtenues par le PMC sans utilisation de contraintes sur les sorties. Les termes sont représentés par leur mesure  $U$  (gauche) et leur mesure  $U$  plus utilisation d'informations syntaxiques (droite).

#### 6.4.4.2 Comparaison des modèles

Afin d'évaluer l'apport de nos modèles par rapport aux modèles classiques existants, nous avons mené d'autres expériences avec les modèles (6.14), (6.15) et le MMC de base (6.16), noté  $W_{MMC}$  [AMI00a].

	<b>Surlignage</b>		<b>Extraction</b>		
	% bonne classification		% bonne classification		
	<i>Pertinent</i>	<i>Moyenne</i>	<i>Position</i>	<i>Personne</i>	<i>Moyenne</i>
	<i>I-R-I</i>		<i>I-&lt;Per-Pos-I&gt;</i>		
$U_{PMC}$	81.06	<b>82.45</b>	61.58	<b>38.34</b>	<b>70.20</b>
$U_{MMC}$	<b>85.15</b>	62.73	<b>87.49</b>	12.29	57.37
$W_{MMC}$	45.29	60.02	55.49	10.31	56.34
	$\langle I/R^{(3)} \rangle$		$\langle I/Per^{(3)} Pos^{(3)} \rangle$		
$U_{PMC}$	85.32	<b>78.9</b>	74.31	<b>48.09</b>	<b>73.51</b>
$U_{MMC}$	<b>86.57</b>	71.02	<b>75.49</b>	22.31	66.17
$W_{MMC}$	41.13	67.52	43.44	0	65.69

Tableau 4. Les Performances des modèles  $U_{PMC}$ ,  $U_{MMC}$  et  $W_{MMC}$  pour le surlignage et l'extraction de surface pour quatre grammaires. Dans le cas de surlignage, la moyenne est sur les classes pertinent et non-pertinent tandis que dans le cas de l'extraction elle est sur les classes Position, Personne and non-pertinent. Les meilleures performances sont en gras.

Le tableau 4 résume ces expériences et donne le pourcentage de bonne classification (PBC) de ces trois modèles. Dans une première expérience nous utilisons respectivement les grammaires  $I-R-I$  et  $I-\langle Per-Pos-I \rangle$  pour le surlignage et l'extraction. Pour cette expérience,  $U_{PMC}$  et  $W_{MMC}$  donnent une nette amélioration sur  $W_{MMC}$  mais il n'est pas clair lequel de ces deux modèles est le meilleur. Les performances moyennes sont néanmoins meilleures pour  $U_{PMC}$ .

Dans une deuxième expérience, nous avons mené les mêmes tests que précédemment mais cette fois avec une contrainte de durée minimum. Pour le surlignage, les performances de  $I$  et  $R$  sont en ordre inverse comparé à la première expérience. Pour l'extraction, la grammaire  $\langle I/Per^{(3)}|Pos^{(3)} \rangle$  donne une nette amélioration de la performance : le pourcentage des étiquettes correctes pour différentes classes est assez homogène tandis que les performances globales sont meilleures. Ceci montre que le modèle exploite avec succès l'introduction de la connaissance du domaine.

Les performances de la classe Personne sont beaucoup moins élevées que celles de la classe Position pour tous les modèles. Ceci s'explique car, dans le corpus, la classe Personne a moins d'occurrences de termes (8%) que la classe Position (32%) et les mots de cette classe sont très souvent étiquetés avec la classe Position (dans notre cas le modèle  $W_{MMC}$  classe toutes les occurrences de la classe Personne en classe Position).

Pour tous les modèles, les probabilités de transition influencent grandement sur les performances et pourront changer la balance entre les classes Personne et Position (ces probabilités peuvent être initialisées efficacement avec la méthode de cross-validation, mais cela n'est pas le cas ici). Dans toutes les expériences, les meilleures performances en moyenne sont obtenues avec le modèle  $U_{PMC}$ .

La Figure 39 montre les courbes de précision-rappel pour les modèles  $U_{PMC}$  et  $W_{MMC}$  dans le cas de surlignage en utilisant la grammaire  $\langle I/R^{(3)} \rangle$ .

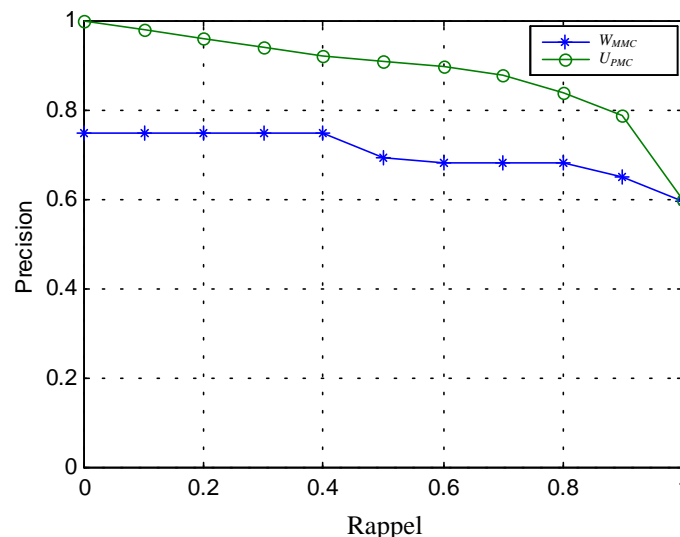


Figure 39. Les courbes de précision-rappel obtenues avec les modèles  $U_{PMC}$  (ligne du haut) and  $W_{MMC}$  (ligne du bas) pour la tâche de surlignage.

La figure 40, montre la même expérience pour le cas de l'extraction. Les courbes de précision-rappel sont obtenues pour les classes Position (Figure 40 gauche) et Personne

(Figure 40 droite). Dans tous les cas, le modèle  $U_{PMC}$  surpasse le modèle de base  $W_{MMC}$ . Pour les deux modèles, les performances de la classe Personne sont moins élevées.

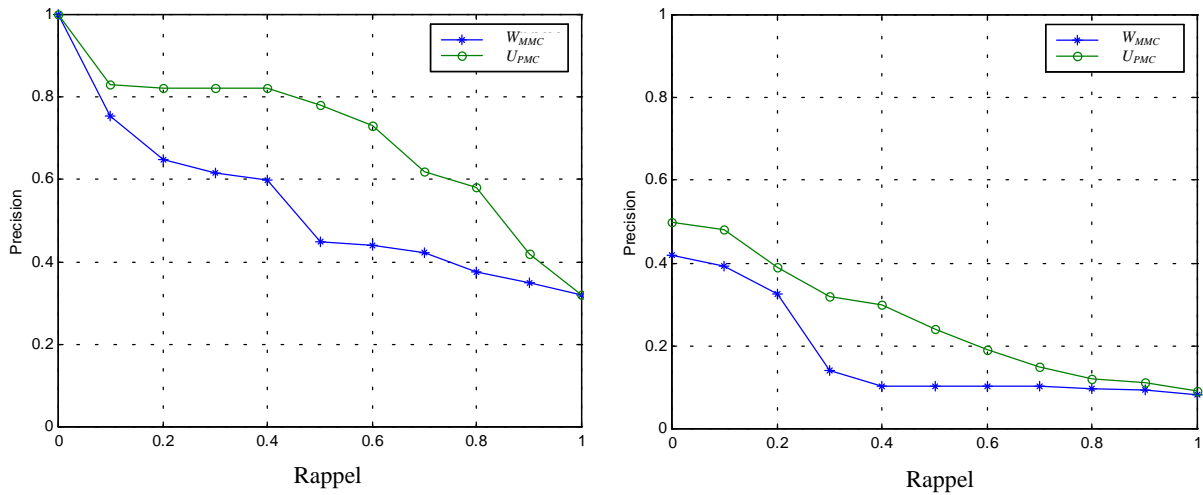


Figure 40. Les courbes de Précision-Rappel obtenues avec les modèles  $U_{PMC}$  et  $W_{MMC}$  pour l'extraction des classes Position (gauche) et Personne (droite).

## 6.4.5 Validation

Afin de valider les résultats obtenus précédemment, nous avons cherché à déterminer la précision des performances globales de  $U_{PMC}$  ainsi que des intervalles de confiance pour ces performances [AMI00b], pour cela, nous avons utilisé la méthode statistique de Bootstrap. [TIB96] utilise cette méthode pour estimer l'écart type des performances d'un PMC. Pour cela, nous avons construit aléatoirement 40 bases de données par un tirage avec remise du corpus initial. Pour chaque base, nous avons réalisé les expériences de surlignage et d'extraction avec le modèle  $U_{PMC}$ , en considérant les deux représentations ( $U$ ) et ( $U$  + informations syntaxiques). Dans cette partie, notre but est de valider les performances obtenues précédemment en cherchant.

### 6.4.5.1 Validation des performances

Pour évaluer nos performances, nous avons choisi la méthode de Bootstrap [TIB96], qui estime la précision avec laquelle la moyenne des performances est calculée. Un exemple Bootstrap est obtenu en échantillonnant  $B$  fois la base initiale  $D_i$ , par un tirage avec remise  $x \rightarrow (D_i^{*1}, D_i^{*2}, \dots, D_i^{*B})$ . On calcule alors pour chaque échantillon  $D_i^{*b}$  la valeur statistique qui nous intéresse  $s(D_i^{*b})$ , des performances globales dans notre cas. L'estimation de l'erreur de Bootstrap standard est la déviation standard des échantillons bootstrapés. L'algorithme de Bootstrap demande alors le calcul des paramètres  $s(D_i^{*b})$  de l'ensemble des points  $D_i^{*b}$  et l'évaluation de la variance de ces paramètres autour de leur moyenne. Dans notre cas nous avons mis en œuvre cet algorithme de la façon suivante :

Générer  $B=40$  exemples par un tirage avec remise de la base initiale,  
 Lancer le PMC sur ces bases,  
 Evaluer les performances globales et locales de chaque base,  
 Estimer l'écart type de ces performances en calculant leur variance.

Algorithme 17. Algorithme de Bootstrap

Le tableau 5 montre les résultats de Bootstrap obtenu avec le  $U_{PMC}$ . Dans le cas de surlignage, lorsque l'on utilise, ou pas, des étiquettes syntaxiques pour la représentation des termes, et une grammaire pour la décision du modèle, les écarts types de la performance globale et de la classe pertinente sont faibles.

Surlignage	Performance globale		Classe Pertinente	
	Moyenne	Ecart type	Moyenne	Ecart type
U Sans grammaire	80.23	0.26	80.52	0.51
U + Etiquette Sans grammaire	80.49	0.3	80.49	0.3
U <I/R <sup>(3)</sup> >	80.64	0.002	80.34	0.6
U + Etiquette <I/R <sup>(3)</sup> >	<b>81.64</b>	0.003	<b>81.35</b>	0.55

Tableau 5. Performance Moyenne et Ecart type du modèle  $U_{PMC}$  obtenue par Bootstrap dans le cas de surlignage avec  $B = 40$ .

Ceci montre d'abord que le corpus initial est homogène, et qu'ensuite le modèle est assez robuste pour différencier entre les phrases de différentes classes. Ces expériences montrent que le  $U_{PMC}$  est un modèle valide pour trouver une séquence pertinente à l'intérieur d'un paragraphe.

6.4.5.2 Intervalle de Confiance t-Bootstrap par classe

Le calcul de la performance moyenne ne donne pas de précision quant à l'étalement des performances autour de cette valeur. Il est donc intéressant d'estimer un intervalle de confiance pour les performances calculées précédemment. En se basant sur l'échantillon des données on peut obtenir par la méthode de Bootstrap des intervalles de confiance, appelés *t-Bootstrap*, [EFR93].

Cette procédure est la suivante :

En générant  $B$  échantillons de bootstrap ( $D_i^{*1}, \dots, D_i^{*B}$ ) on calcule  $Z^*(b) = (\hat{\theta}^*(b) - \hat{\theta}) / \hat{sê}$  pour chaque échantillon, où  $\hat{\theta}^*(b)$  est la valeur à estimer pour le  $b$  ième échantillon et,  $\hat{\theta}$  et  $\hat{sê}$  sont respectivement la moyenne et l'écart type des valeurs  $\hat{\theta}^*$ . Les bornes de l'intervalle de confiance à  $\alpha\%$  sont le  $\alpha^{ém}$  fractile de  $Z^*(b)$ , estimé par la valeur  $\hat{\omega}^{(\alpha)}$  tel que  $|Z^*(b)| \leq \hat{\omega}^{(\alpha)} \mid B = \alpha$ , le paramètre  $\theta$ , dans notre cas, est la valeur moyenne des performances.

L'intervalle de confiance à  $\alpha\%$  donnée par la méthode *t-Bootstrap* est finalement :

$$\left[ \hat{\theta} - \hat{\omega}^{(1-\alpha)}.se, \hat{\theta} - \hat{\omega}^{(\alpha)}.se \right]$$

La figure 41 montre les intervalles de confiance à 95% calculées pour les performances globales, ainsi que pour les classes *Personne* et *Position* obtenues par le  $U_{PMC}$ , dans le cas de l'extraction, pour les deux représentations de texte. On remarque que la contrainte grammaticale  $\langle I/R^{(3)} \rangle$  imposée à la sortie du  $U_{PMC}$ , améliore les performances du système.

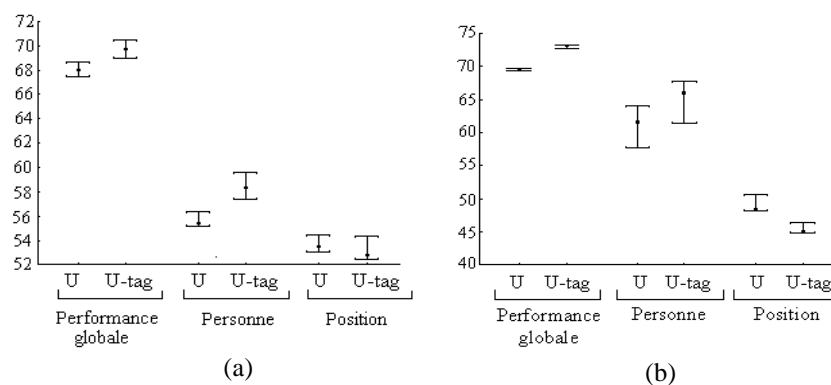


Figure 41. Intervalle de confiance t-bootstrap à 95%, calculer pour la tâche de l'Extraction sans (a) et avec (b) utilisation de grammaire  $\langle I/R^{(3)} \rangle$  avec le modèle  $U_{PMC}$ .

## 6.5 Conclusion

Nous avons décrit dans ce chapitre un formalisme original unifié pour l'analyse automatique de séquences de mots dans le cadre de tâches d'accès à l'information avec des requêtes fermées. De ce formalisme général, nous avons dérivé deux modèles probabilistes correspondant à différents types d'information recherchés. Ces modèles opèrent sur des espaces de représentation des termes de faible dimension qui constituent également un apport de notre travail. Nous les avons implémenté en utilisant des estimateurs de maximum de vraisemblance pour des densités (ex. MMC) ou des estimateurs de probabilité à posteriori (ex. PMC). Afin d'améliorer les performances, nous avons introduit des contraintes grammaticales et nous avons montré que cela permettait d'améliorer effectivement sensiblement les systèmes.

Nous avons illustré le potentiel de nos modèles sur deux exemples d'application : l'extraction de l'information de surface et le surlignage. Les comportements et les performances de nos systèmes ont été comparés avec un modèle classique qui implémente un classifieur naïve Bayes couplé avec l'algorithme d'alignement de Viterbi. Pour chacune des applications traitées, nous avons montré l'intérêt d'utiliser des informations morpho-syntaxiques dans la représentation des termes et des



grammaires en sortie de ces modèles. Notamment, notre représentation simple donne de meilleurs résultats avec des modèles de séquences que le modèle Naïve Bayes utilisant comme entrée la distribution unigramme sur les termes.

Ce type de modèle ouvre des perspectives pour des applications d'extraction d'information ou de segmentation, en particulier dans des applications où le développement de systèmes manuels est trop coûteux.

## 6.6 Bibliographie

- [AMI99a] Amini M.-R., Zaragoza H., Gallinari P. Sequence Models For Automatic Highlighting and Surface Information Extraction. *Information Retrieval Special Group, BCG-IRSG'99*, pp. 85--91, 1999.
- [AMI99b] Amini M.-R., Zaragoza H., Gallinari P. Stochastic Models for Surface Information Extraction in Texts, *International Conference on Artificial Neural Networks, ICANN'99*, pp.892--897, 1999.
- [AMI00a] Amini M.-R., Zaragoza H., Gallinari P. Learning for Sequence Extraction Tasks, *RIAO'2000, CID, Paris*, pp. 476--490, 2000.
- [AMI00b] Amini M.-R., Zaragoza H., Gallinari P. Utilisation des Réseaux de Neurones pour l'Analyse de Séquences dans les Textes, *Conférence d'Apprentissage CAp'2000, Hermès Science*, pp. 21--30, 2000.
- [AND92] Anderson E. The Statistical Analysis of Categorical Data, *Springer*, 1992.
- [BIK99] Bikel D., Schwartz R., Weischedel R.M. An algorithm that Learns what's in a Name, *Machine learning, ML'99*, 34, pp. 211--231, 1999.
- [BIS95] Bishop C.M. *Neural networks for Pattern Recognition*, Oxford : Clarendon Press. 1995.
- [BON96] Bonnländer, Weigend A.S. Selecting Input Variables Using Mutual Information and Nonparametric Density Evaluation, *in Proceedings of the International Symposium on Artificial Neural Networks, ISANN'96*, pp. 42--50, 1996.
- [CHA93] Charniak E., Hendrickson C., Jacobson N., Perkowitz M. Equations for part of speech tagging, *11<sup>th</sup> Nat. Conf. On AI, AAAI*, pp.784--789, 1993.
- [EFR93] Efron B., Tibshirani R. An Introduction to the Bootstrap, *Editions Chapman & Hall*, 1993.
- [FRE99] Freitag D., McCallum A. Information extraction with HMMs and shrinkage, *AAAI-99 workshop on machine learning for information extraction*, 1999.
- [GAL00] Gallinari P., Zaragoza H., Amini M.-R. Apprentissage et Données Textuelles. *Ecole Modulad – SFdS, Livre à praître*, 2000.
- [MIT94] Mittendorf E. and Schauble P. Document Passage Retrieval Based on Hidden Markov Models, *ACM SIGIR'94*, pp. 318--327, 1994.
- [MUC6] MUC-6, *Proceedings of the sixth Message Understanding Conference*, Morgan Kaufmann, Publishers, 1996.
- [SCH94] Schmid H. Probabilistic Part-of-Speech Tagging Using Decision Trees. *In Proc. of Int. Conference on New Methods in Language Processing*, Manchester, UK, 1994.
- [TIB96] Tibshirani R. A comparison of Some Error Estimates for Neural Networks Models. *Neural Computation*, pp. 153--163, vol. 8, n° 3, 1996.

[VAP82] Vapnik V. Estimation of Dependencies Based on Empirical Data, *Springer-Verlag*, 1982.

[ZAR99] Zaragoza H. Modèles Dynamiques d'Apprentissage Numériques Pour l'accès à l'Information, *Thèse de Doctorat en Informatique*, Université de Pierre et Marie Curie, Paris 6, 1999.



## Chapitre 7

### Apprentissage interactif, semi-supervisé et non-supervisé pour le résumé de texte

**Résumé.** Dans ce chapitre nous proposons d'introduire de l'apprentissage pour faire du résumé automatique de texte. En partant d'un système de base, classique pour réaliser des résumés par extraction de phrases pertinentes, nous montrons comment l'introduction de l'apprentissage permet d'obtenir des systèmes de résumé plus performants. Nous commençons par décrire un système d'apprentissage interactif où l'utilisateur donne au système des informations sur la pertinence des phrases qu'il lui propose. Ce système bien que plus efficace que le système de base, demande un investissement de la part de l'utilisateur qui peut ne pas convenir à certains modes d'utilisation. Nous proposons une deuxième approche, basée sur de l'apprentissage semi-supervisé et non-supervisé, qui pallie cet inconvénient et est entièrement automatique. Cette approche repose sur l'utilisation d'un classifieur qui peut être choisi dans n'importe laquelle des familles connues de classifieurs. Nous analysons l'algorithme proposé dans deux cas simples : celui d'un classifieur linéaire et celui d'une discrimination logistique. Pour chaque cas, nous proposons une preuve de convergence de l'algorithme et montrons qu'il s'agit d'instances de l'algorithme CEM. Les deux algorithmes correspondants sont respectivement appelés *CEM-régression* et *CEM-logistique*. L'originalité de l'approche est qu'elle permet de faire de l'apprentissage non-supervisé et semi-supervisé à partir de modèles discriminants, ce qui lui assure une bonne robustesse et une grande facilité de mise en œuvre. L'évaluation de ces modèles appliqués au résumé de texte a été faite sur la base Reuters.

**Mots clés :** Apprentissage interactif, apprentissage semi-supervisé, apprentissage non-supervisé, résumé de texte, CEM-régression, CEM-logistique, base Reuters.

## 7.1 Introduction

Le résumé automatique permet de présenter à l'utilisateur une information de qualité qui caractérise un texte, sous une forme extrêmement condensée et qui peut être facilement appréhendée. Nous avons vu dans le chapitre 5 que devant la difficulté de réalisation de résumés synthétiques ressemblant à des résumés humains, les travaux s'étaient orientés vers le résumé par extraction de phrases pertinentes par rapport à une requête.

La plupart des méthodes de résumé automatique procèdent en triant les phrases d'un document par rapport à une mesure de similarité avec la requête et en sélectionnant celles ayant les plus grands scores. Récemment, différents auteurs ont commencé à explorer l'utilisation de techniques d'apprentissage automatique pour effectuer des résumés. Par rapport aux méthodes reposant sur des mesures de similarité, ces techniques permettent de s'adapter au corpus traité ou aux demandes particulières de l'utilisateur. Toutes les approches proposées jusqu'à aujourd'hui reposent sur de l'apprentissage supervisé, ce qui pour effectuer du résumé par sélection de phrases nécessite l'étiquetage de ces dernières. Une des difficultés est alors que l'étiquetage d'une large base de documents est très coûteuse et de plus, trier les phrases est intrinsèquement difficile du fait qu'il n'existe pas de caractérisation de ce qu'est un bon résumé.

Nous allons présenter deux nouvelles approches qui facilitent l'apprentissage des systèmes pour cette tâche. La première est basée sur l'apprentissage interactif : le système utilise le jugement d'un utilisateur pour corriger ses erreurs de décisions. La deuxième approche est basée sur l'apprentissage non-supervisé et semi-supervisé et elle va plus loin dans le sens de l'automatisation car le système n'a pas besoin de l'interaction utilisateur pour apprendre. Les deux approches reposent sur l'utilisation d'un classifieur qui sert à fournir un score de pertinence pour chacune des phrases du texte. Nos expériences ont montré que de bonnes performances étaient obtenues avec des classifieurs simples et qu'il était inutile pour cette application d'utiliser des systèmes de classification trop complexes. Nous avons alors analysé deux modèles correspondant à ceux que nous utilisons en pratique et appelés respectivement *CEM-régression* et *CEM-logistique*.

- Le premier modèle est linéaire, nous montrons que sous une hypothèse de populations gaussiennes (phrases pertinentes et non-pertinentes), il s'agit d'une instanciation de l'algorithme CEM.
- Le deuxième modèle est plus puissant dans le sens où nous n'avons pas besoin de faire beaucoup d'hypothèses sur les distributions de points : la seule hypothèse nécessaire est celle des modèles logistiques. Là aussi nous montrons le lien avec l'algorithme CEM. Ce modèle s'implémente facilement avec un neurone possédant comme fonction de transfert une fonction sigmoïde.

Ces deux modèles adoptent une approche discriminante au lieu d'estimer des densités conditionnelles comme la plupart des méthodes basées sur des techniques non-

supervisées et semi-supervisées. Ils maximisent tous les deux la vraisemblance de classification. Pour chaque cas, nous donnons une preuve de convergence.

Ce chapitre est organisé de la façon suivante, dans la première partie nous allons présenter le système interactif (section 2) et dans la deuxième partie nous allons présenter nos modèles *CEM-régression* et *CEM-logistique* (section 3). Nous donnons notre conclusion dans la section 7.

## 7.2 Modèle interactif pour le résumé de textes

Dans [Ami00], nous avons présenté un système de résumé basé sur l'interaction utilisateur pour l'apprentissage des paramètres du système de classification qui permet d'attribuer un score aux phrases. Ce système procède en deux étapes : d'abord il extrait les phrases les plus pertinentes d'un document par rapport à une requête en utilisant la pondération classique *tf-idf*, il apprend ensuite grâce à l'interaction utilisateur afin d'améliorer ses performances et de s'adapter aux désirs de l'utilisateur. Nous allons présenter dans cette partie ces deux étapes.

### 7.2.1 Un système de base pour effectuer du résumé automatique : extraction de phrases en utilisant des mesures de similarité

Les systèmes d'extraction de phrases classiques utilisent généralement des mesures de similarités entre les phrases (ou parfois des paragraphes) et une requête. Les phrases les plus pertinentes sont ensuite sélectionnées en comparant les scores des phrases du document avec un seuil donné. La différence majeure entre les systèmes existants est la représentation de l'information textuelle et les mesures de similarités qu'ils utilisent.

Généralement, des caractéristiques statistiques et/ou linguistiques sont utilisées, afin d'encoder les phrases d'un document et de la requête en un vecteur de taille fixe. Des similarités simples (du type cosinus) sont ensuite calculées.

Nous partirons du travail de [Kna94] qui a proposé une technique de ce type pour l'extraction de phrases pertinentes pour une requête. Ils utilisent une représentation *tf-idf* et calculent la similarité entre une phrase  $\varphi_k$  et une requête  $q$  par :

$$Sim_1(q, \varphi_k) = \sum_{w_i \in \varphi_k, q} tf(w_i, q) \cdot tf(w_i, \varphi_k) \cdot \left( 1 - \frac{\log(df(w_i) + 1)}{\log(n + 1)} \right)^2 \quad (7.1)$$

où  $tf(w, x)$  est la fréquence du terme  $w$  dans  $x$  ( $q$  ou  $\varphi_k$ ),  $df(w)$  est le nombre des documents contenant  $w$  dans la base et  $n$  le cardinal de la collection de documents. Les phrases  $\varphi_k$  et la requête  $q$  sont pré-traitées afin d'enlever les mots de l'ante-dictionnaire et la méthode de lemmatisation de Porter [POR80] est ensuite appliquée sur les mots restants. Pour chaque document on estime alors un seuil afin de déterminer les phrases les plus pertinentes par rapport à la requête.

Notre approche pour l'étape d'extraction de phrases est une variation de cette méthode, où la requête est d'abord enrichie avant le calcul de la similarité. Comme les phrases et les requêtes peuvent être très courtes, cette méthode permet de calculer des mesures de similarités plus correctes. Dans nos expériences, l'enrichissement de la requête est apparue comme très importante.

Pour étendre la requête nous avons procédé en deux étapes :

1. La requête est d'abord étendue grâce à un thesaurus général - Wordnet dans notre cas.
2. Les phrases les plus pertinentes dans le texte sont ensuite extraites et les mots les plus fréquents de ces phrases sont inclus dans la requête.

Ce procédé peut être réitéré. La mesure de similarité que nous avons considérée est :

$$Sim_2(q, \varphi_k) = \sum_{w_i \in \varphi_k, q} \bar{tf}(w_i, q) \cdot tf(w_i, \varphi_k) \cdot \left( 1 - \frac{\log(df(w_i) + 1)}{\log(n + 1)} \right)^2 \quad (7.2)$$

Où,  $\bar{tf}(w, q)$  est le nombre de termes dans la classe «sémantique» de  $w$  dans la requête  $q$ . L'expansion de la requête a été utilisée avec succès pendant plusieurs années par la communauté recherche d'information [Xu96].

Ce système sera utilisé comme un système de base, afin de mesurer l'impact de l'apprentissage et de l'interaction. Bien qu'il soit très simple, il est bien adapté pour le résumé de texte basé sur l'extraction de phrases. Par exemple, [Zec96] a utilisé ce système qui opère simplement sur la fréquence des mots, dans le contexte du résumé générique, et il a montré qu'il donnait de bons résultats en le comparant à l'extraction de phrases faite par des experts humains.

### 7.2.2 Apprentissage interactif

Les méthodes basées sur des mesures de similarités ont des limitations intrinsèques : elles reposent sur des mesures et des caractéristiques prédéfinis, elles sont conçues pour des tâches génériques, leur adaptation à un corpus spécifique, aux différents genres de documents ou à une communauté d'utilisateurs spécifique doit être réglée manuellement.

Plusieurs auteurs ont récemment proposé l'utilisation de techniques d'apprentissage pour améliorer les qualités ou l'adaptabilité des systèmes de résumés [Gol99, Jin99]. Ceci est d'autant plus important dans le contexte d'Internet où les différents types de documents et les demandes d'utilisateurs peuvent varier considérablement. L'apprentissage permet d'exploiter les caractéristiques de corpus et l'interaction utilisateurs. Nous allons proposer ci-dessous une technique qui prend en compte ces deux aspects et qui permet d'améliorer significativement la qualité des phrases extraites. Dans l'approche proposée, l'apprentissage est basé sur l'interaction utilisateur. La requête d'un utilisateur est d'abord traitée par le système de base décrit en section 7.2.1 et les  $N$  phrases les plus pertinentes sont ensuite présentées à l'utilisateur (figure 42). L'utilisateur peut ensuite interagir avec le système en indiquant dans chaque document les phrases qu'il juge pertinentes ou non-pertinentes pour sa requête (figure 43).

L'apprentissage basé sur cette interaction opère ensuite à deux niveaux :

1. L'apprentissage peut être utilisé afin de changer la représentation de la requête comme dans une interaction classique, nous avons utilisé pour cela une approche similaire à [Roc71]. Ceci permet d'étendre la requête en accord avec le jugement de l'utilisateur et d'affiner l'expansion faite durant la première étape (cf. chapitre 4, section 4.3.4.3).



2. Un classifieur est entraîné en accord avec la pertinence des phrases d'un document pour le résumé. Posons  $P_q$  la variable binaire qui est égale à 1 si la phrase est pertinente par rapport à la requête  $q$  et 0 sinon. Les phrases pertinentes sont celles sélectionnées par le système de base et dont la pertinence n'a pas été changée par l'utilisateur ainsi que les phrases jugées pertinentes par l'utilisateur. On définit de la même manière les phrases non-pertinentes. Le classifieur sera entraîné à estimer  $P(P_q/\varphi)$ , la probabilité a posteriori de pertinence par rapport à la requête sachant la phrase. L'utilisation d'un classifieur permet d'adapter la frontière de décision au corpus et aux besoins de l'utilisateur.

Le classifieur que nous avons employé opère sur des phrases qui sont codées comme des suites de termes. Il procède du même esprit que les classifieurs que nous avons utilisés dans le chapitre 6 pour l'extraction d'information. Nous le décrivons ci-dessous.

### *Le classifieur*

Pour pouvoir calculer  $p(P_q/\varphi)$  où  $P_q$  est la variable binaire indicatrice de la pertinence de la phrase  $\varphi$ , nous devons faire quelques hypothèses simplificatrices. Soit  $x_1^{|\varphi|} = x_1 \dots x_\varphi$  une séquence de vecteurs caractéristiques de termes, représentant une phrase  $\varphi$  de taille  $|\varphi|$ . Dans notre cas, comme nous allons le voir à la section 7.5, les  $x_i$  sont des vecteurs de taille 4.

Sous l'hypothèse d'indépendance des vecteurs caractéristiques :

$$p(x_1^{|\varphi|}) = \prod_i p(x_i).$$

Nous pouvons montrer facilement que :

$$p(P_q / \varphi) = \prod_{x_i \in \varphi} p(P_q / x_i) \quad (7.3)$$

L'équation (7.3) sera utilisée pour calculer le score d'une phrase. En pratique nous avons trouvé la décomposition suivante plus efficace :

$$p(P_q / \varphi) = \prod_{x_i \in \varphi} p(P_q / x_{i-2, i+2}) \quad (7.4)$$

Qui se justifie en apportant une modification minimale à l'équation (7.3). Comme pour l'extraction de séquence présentée au chapitre 6 nous considérons un contexte local du terme  $x_i$  (une fenêtre de taille 2 autour de  $x_i$ ) pour le calcul du score de pertinence. La taille de la fenêtre a été déterminée expérimentalement. Pour l'implémentation de ce classifieur nous avons utilisé un neurone sigmoïde. L'hypothèse d'indépendance est très forte, sa seule justification est qu'elle rend possible le calcul de la probabilité a posteriori et fournit un score pour trier les phrases. D'autres hypothèses plus faibles peuvent être utilisées, mais en pratique, l'équation (7.4) s'est révélée satisfaisante.

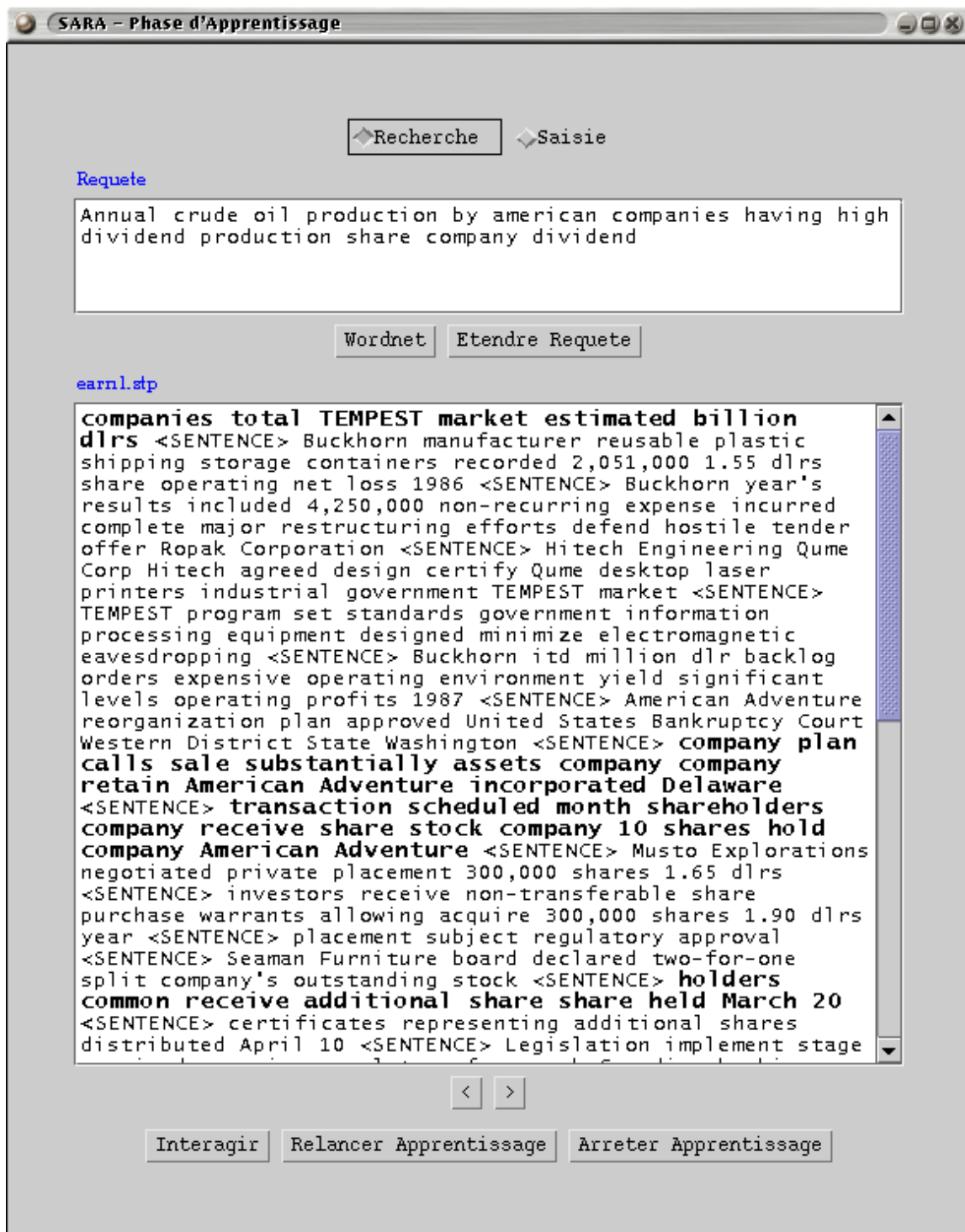


Figure 42. Projection de la requête sur les phrases des documents, les phrases dont la similarité avec la requête est supérieure à un seuil calculé, sont indiquées en gras, elles seront proposées à l'utilisateur comme pertinentes.

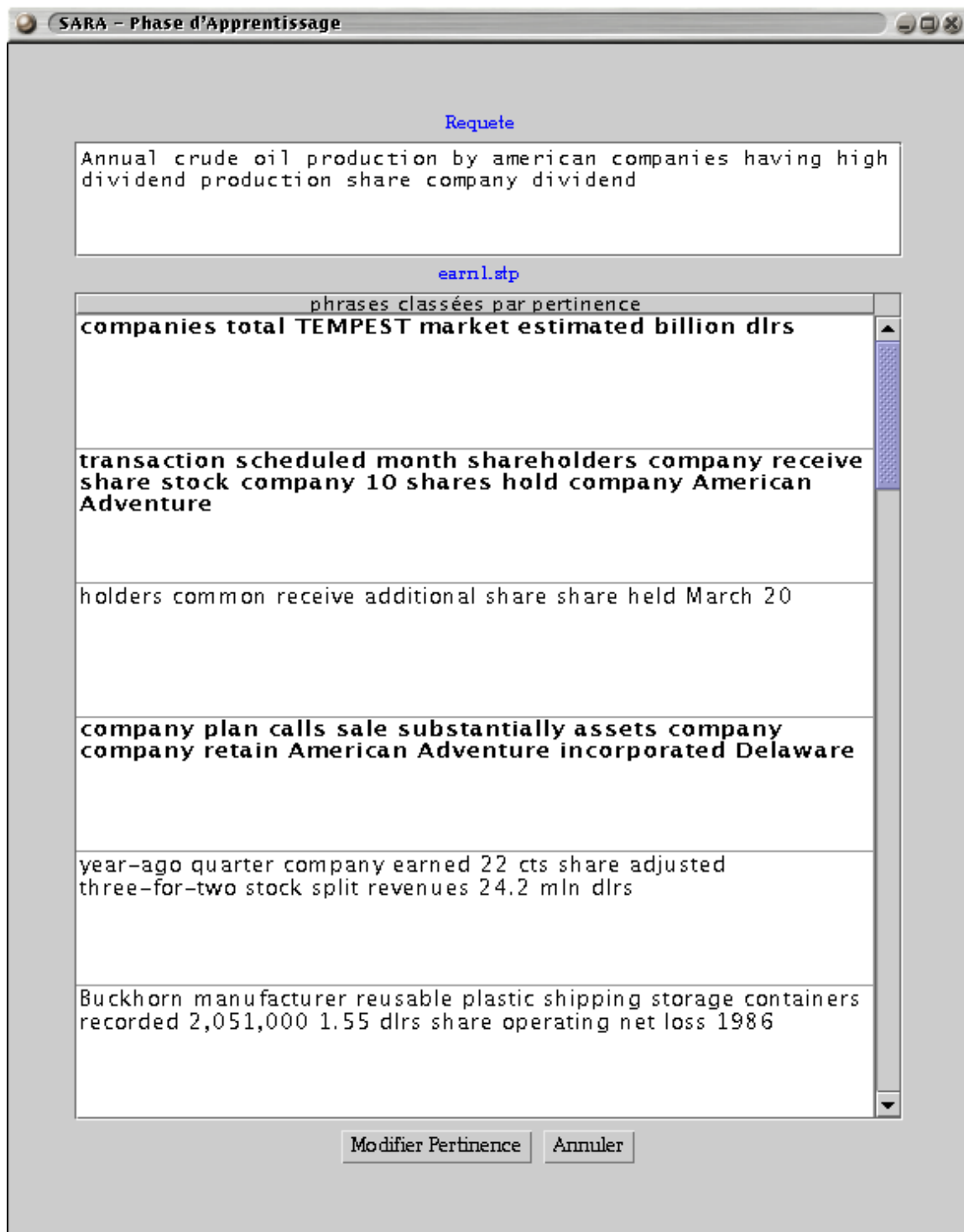


Figure 43. Les phrases jugées pertinentes à une étape  $j$  sont présentées à l'utilisateur dans l'ordre de leur pertinence et celui-ci interagit avec le système en indiquant pour certaines d'entre elles quelles sont celles qu'il juge pertinentes ou non-pertinentes.

### 7.3 Modèle automatique basé sur l'apprentissage non-supervisé et semi-supervisé pour le résumé

L'interaction utilisateur, constitue un enrichissement du modèle de base et permet d'augmenter ses performances sensiblement, comme nous le verrons ultérieurement. Cette approche peut se révéler pertinente quand l'utilisateur est prêt à passer du temps pour obtenir des résumés de meilleure qualité ou personnalisés. Elle est prohibitive sinon.

Aussi, nous avons exploré des techniques à base d'apprentissage non-supervisé et semi-supervisé pour automatiser complètement les méthodes de résumé de textes.

Ces méthodes peuvent aussi bien s'acquitter des tâches de résumé générique que de celles basées sur les requêtes utilisateurs. Elles permettent de tirer avantage d'une large collection de documents non étiquetés, i.e. pour lesquels on ne dispose pas de résumé. Les travaux antérieurs sur l'application de méthodes d'apprentissage au résumé de textes, vus au chapitre 5, ont débuté en 1995 et reposent tous sur l'apprentissage supervisé [Kup95, Teu97, Man98, Gol99, Chu00]. Avec une telle approche, on a besoin d'une base d'apprentissage et de leur résumés associés, qui sont utilisés pour étiqueter les phrases des documents comme pertinentes ou non-pertinentes pour le résumé. Après le cycle d'apprentissage, ces systèmes opèrent en étiquetant les phrases des nouveaux documents par rapport à leur score de pertinence avec la requête. L'étiquetage de grandes bases au niveau des phrases est clairement prohibitif et s'applique mal au cas de requêtes non génériques.

Du point de vue apprentissage, la tâche de résumé est typiquement une tâche pour laquelle il existe de grandes quantités de données non étiquetées (tous les textes disponibles dans les différents corpus) et où les données étiquetées sont très chères. C'est par définition un cadre idéal pour l'apprentissage semi-supervisé ou non-supervisé.

Ces techniques n'ont pas, à notre connaissance, été employées dans le cadre du résumé. En recherche d'information, elles ont principalement été utilisées pour faire de la classification de texte. La plupart des travaux dans ce cadre partent d'une approche non-supervisée et proposent :

- d'adapter l'algorithme EM pour prendre en compte des données étiquetées et non étiquetées,
- d'appliquer une estimation de maximum de vraisemblance.

Les modèles utilisés sont la plupart du temps des mélanges de gaussiennes ou dans le cas discret des mélanges de lois multinomiales. Différents auteurs ont observé qu'en pratique ces hypothèses sont trop restrictives et ont proposé des modèles plus complexes pour s'adapter au traitement de données réelles.

L'approche que nous proposons part de techniques d'apprentissage supervisé. Elle est proche dans l'esprit de l'algorithme de décision dirigée vu dans le chapitre 3 section 3.3.1. Nous partons d'une partition initiale des phrases qui peut être réalisée soit par un algorithme de base tel que celui qui est présenté en section 7.2.1, dans le cas non-supervisé, soit par un algorithme de discrimination entraîné sur des données étiquetées dans le cas semi-supervisé, soit encore par une combinaison des deux. Ce premier

étiquetage est ensuite amélioré de façon itérative en utilisant un classifieur qui utilise ses propres sorties et la confiance qui leur est associée pour re-étiqueter les phrases.

Cet algorithme est décrit de façon informelle ci-dessous (Algorithme 18) pour le cas non-supervisé.

---

**Entrée** : un ensemble de phrases non étiquetées  $D_u = (x_1, \dots, x_m)$ ,

Donner un partitionnement initial des phrases  $P^{(0)}$  par un système de base, notons  $\tilde{T}^{(0)}$  l'étiquetage des phrases de  $D_u$  issu de ce partitionnement.

Pour  $j = 0$  jusqu'à ce qu'il n'y ait plus de changement sur l'étiquetage des  $x_i$

{

① Apprendre un classifieur de base sur  $(D_u, \tilde{T}^{(j)})$  et donner une estimation de la probabilité a posteriori de la classe des  $x_i$  ( $x_i \in D_u$ ).

② En déduire un nouvel étiquetage des phrases de  $D_u$  :  $\tilde{T}^{(j+1)}$

}

**Sortie** : L'étiquetage des phrases

---

#### Algorithme 18. Algorithme informel pour l'apprentissage non-supervisé

Dans le cas semi-supervisé, on modifiera l'algorithme de façon à prendre en compte les données étiquetées. Par rapport à l'algorithme 18, celles ci seront utilisées

- pour l'initialisation du classifieur : le classifieur est d'abord entraîné à partir de ces données.
- pour l'étape 1, les étiquettes de  $D_l$  sont connues et sont fixes, ces données étiquetées servent à entraîner le classifieur conjointement aux données non étiquetées.
- Pour l'étape 2, l'étiquetage des phrases de  $D_l$  est inchangé par cette étape

La version non supervisée de ces algorithmes pourrait être utilisée pour partitionner (clustering) et la version semi-supervisée pour faire de la discrimination.

L'algorithme présente les avantages des méthodes discriminantes, à savoir qu'il est juste nécessaire d'estimer des probabilités a posteriori  $p(t/x)$  ce qui est bien moins complexe que modéliser la probabilité marginale  $p(x)$  comme dans les approches génératives. Les calculs sont également plus simples. Nous ne faisons pas d'hypothèse sur la forme des données, et n'importe quel classifieur peut être utilisé à l'étape 1 ce qui constitue une des richesses de cette méthode. Nous avons fait des essais avec des réseaux de neurones<sup>12</sup>, et

---

<sup>12</sup> Plus précisément nous avons utilisé un neurone linéaire. Comme algorithme d'apprentissage nous avons appliqué la règle de *Widrow-Hoff* pour minimiser l'erreur quadratique entre les sorties du classifieur et les sorties désirées estimées à l'étape précédente par ce modèle.

des machines à vecteurs supports. Pour la tâche que nous avons abordée et la base de données que nous avons traitée, les meilleures performances sont obtenues avec des classifieurs très simples - des classifieurs linéaires ou logistiques. Notons l'algorithme informel décrit ci-dessus nécessite quelques adaptations pour être opérationnel, elles seront indiquées ultérieurement.

Pour ces différentes raisons, nous avons analysé le comportement de cet algorithme dans le cas de classifieurs linéaires et de classifieurs logistiques. Dans chaque cas, nous avons montré sous certaines hypothèses la convergence de l'algorithme vers un maximum local de la vraisemblance de classification. Nous avons montré pour ces deux classifieurs que l'algorithme est une instance de l'algorithme CEM, dans le cas non-supervisé et dans le cas semi-supervisé. Par contre la mise en œuvre est très différente des utilisations habituelles de l'algorithme CEM. Nous avons également analysé formellement l'utilisation de différents classifieurs dans le cas linéaire. Ces résultats permettent d'éclairer le comportement de l'algorithme dans le cas de classifieurs simples et de s'assurer de sa convergence. En pratique rien n'empêche d'utiliser la même méthode avec des classifieurs plus complexes, mais dans ce cas, il n'existe pas de résultats théoriques.

L'algorithme qui utilise un classifieur linéaire sera dénommé *CEM-régression* par la suite. Sa convergence et le lien avec la décision bayésienne optimale est démontrée sous l'hypothèse où les populations sont normales.

L'algorithme qui emploie un classifieur logistique est nommé *CEM-logistique*. Il est un peu plus général car les hypothèses sont plus faibles.

Les preuves de convergence de nos algorithmes s'appuient sur l'algorithme CEM [CEL92, McL92], qui a été présenté au chapitre 2. Nous allons tout d'abord en décrire une extension pour le cas semi-supervisé (section 7.3.1). Pour prouver la convergence et analyser les algorithmes, nous aurons besoins de quelques résultats de base concernant la décision bayésienne, la régression linéaire et la discrimination logistique - ces résultats sont rappelés en section 7.3.2. Les modèles proprement dits sont introduits à la section 7.3.3. Les expériences sur le résumé sont décrites en section 7.4.

Dans tout ce qui suit, nous nous intéressons à un problème de classification binaire (phrase pertinente ou non-pertinente pour le résumé). Nous allons donc restreindre notre analyse au cas à deux classes.

### 7.3.1 Algorithme CEM dans le cas semi-supervisé<sup>13</sup>

L'algorithme CEM permet de maximiser la vraisemblance de classification (cf. section 2.3.4 et l'annexe). Rappelons que dans le cas classique non-supervisé, cette vraisemblance s'écrit :

$$L_C(P, \pi, \theta) = \sum_{k=1}^c \sum_{i=1}^m t_{ki} \cdot \log(\pi_k \cdot p(x_i / P_k, \theta_k)) \quad (7.5)$$

---

<sup>13</sup> Les notations utilisées sont celles présentées au début de cette thèse

en supposant qu'on dispose de  $m$  données,  $c$  est le nombre de composantes du mélange,  $\pi_j$  les différentes proportions (ou probabilités a priori) et  $p(x/P_k, \theta_k)$  les densités conditionnelles. L'algorithme CEM a été proposé pour faire du clustering et utilisé uniquement avec des modèles génératifs (modèles de densité).

Dans le cas semi-supervisé, les  $t_{ki}$  sont connus pour les  $n$  données étiquetées, le critère devient donc :

$$L_C(P, \pi, \theta) = \sum_{x_i \in D_l} \log(\pi_k \cdot p(x_i / P_k, \theta_k)) + \sum_{x_i \in D_u} \sum_{k=1}^c t_{ki} \cdot \log(\pi_k \cdot p(x_i / P_k, \theta_k)) \quad (7.6)$$

Dans ce cas, on s'appuiera sur les exemples étiquetés pour initialiser et apprendre les paramètres du modèle, l'algorithme est le suivant, nous le formulons dans le cadre de modèles génératifs, comme une extension directe de l'algorithme-CEM :

**Initialisation:** Commencer avec une partition  $P^{(0)}$  initiale sur  $D_l$  définie à partir des étiquettes des exemples.

Pour  $j = 0$  jusqu'à convergence faire

- Etape **E** : Pour les exemples  $x_i \in D_u$ , à partir de la partition courante  $P^{(j)}$  estimer avec les paramètres courants  $\{\pi^{(j)}, \theta^{(j)}\}$ , les probabilités a posteriori d'appartenance aux groupes  $P_k^{(j)}$  pour tout  $k=1, \dots, c$  :

$$E[t_{ki}^{(j)} / x_i; P^{(j)}, \pi^{(j)}, \theta^{(j)}] = \frac{\pi_k^{(j)} \cdot p(x_i / P_k^{(j)}, \theta_k^{(j)})}{\sum_{k=1}^c \pi_k^{(j)} \cdot p(x_i / P_k^{(j)}, \theta_k^{(j)})}$$

- Etape **C** : Assigner à chaque exemple sa classe :  
 Si  $x_i \in D_l$ , l'étiquette de  $x_i$  est  $t_i$ ,  
 Si  $x_i \in D_u$ , l'étiquette de  $x_i$  est  $\hat{t}_k^{(j)}$  où  $k^{(j)}$  est la classe de  $x_i$  à l'étape  $j$ :

$$k^{(j)} = \arg \max_l \frac{\pi_l^{(j)} \cdot p(x_i / P_l^{(j)}, \theta_l^{(j)})}{\sum_{l=1}^c \pi_l^{(j)} \cdot p(x_i / P_l^{(j)}, \theta_l^{(j)})}$$

Noter  $P^{(j+1)}$  la nouvelle partition.

- Etape **M** : Estimer les nouveaux paramètres  $\{\pi^{(j+1)}, \theta^{(j+1)}\}$  qui maximisent  $L_C(P^{(j+1)}, \pi^{(j)}, \theta^{(j)})$  (équation 7.6)

Algorithme 19. Algorithme CEM dans le cas semi-supervisé

### 7.3.2 Fonctions Discriminantes

Dans les sections 7.3.2.1 et 7.3.2.2 nous allons rappeler quelques résultats classiques sur la décision bayésienne et la régression linéaire qui seront utilisés pour analyser notre algorithme d'apprentissage *CEM-régression*.

Dans la section 7.3.2.3 nous allons présenter rapidement la discrimination logistique et l'idée présentée par Anderson pour l'utiliser dans le cas semi-supervisé.

### 7.3.2.1 Règle de décision Bayésienne pour des populations normales

Pour deux populations normales avec une matrice de covariance commune  $\mathcal{N}(\mu_1, \Sigma)$  et  $\mathcal{N}(\mu_2, \Sigma)$  la fonction de décision bayésienne optimale est [Dud73] :

$$g_B(x) = (\mu_1 - \mu_2)^t \cdot \Sigma^{-1} \cdot x + x_0 \quad (7.7)$$

Où  $x_0 = \log \frac{\pi_1}{\pi_2}$  est un seuil. La règle de décision est d'affecter à  $x$  la classe  $C_1$  si  $g_B(x) > 0$  et la classe  $C_2$  sinon (figure 44).

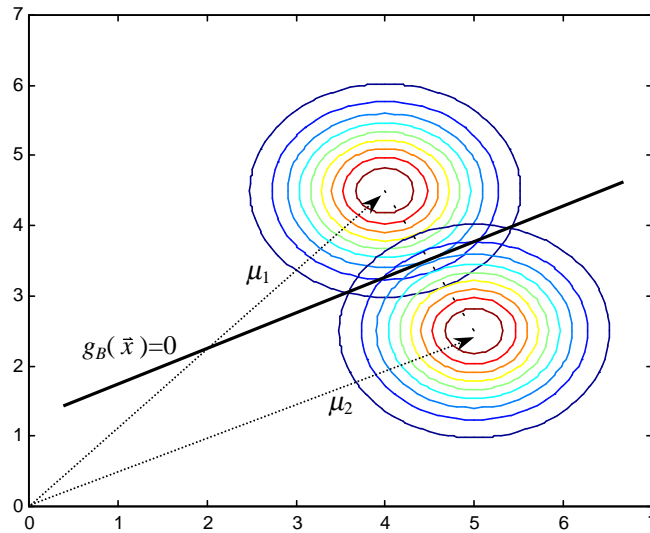


Figure 44. Décision bayésienne pour un problème de classification à deux classes normales de même variance covariance.

### 7.3.2.2 Régression linéaire

Avec les modèles linéaires, la discrimination est souvent traitée comme un problème de régression. Posons  $X$  la matrice dont la  $i^{\text{ème}}$  ligne est le vecteur  $x_i$  et  $Y$  le vecteur colonne désiré dont le  $i^{\text{ème}}$  élément est  $a$  si  $x_i \in C_1$  et  $b$  si  $x_i \in C_2$ . Pour des valeurs de  $a$  et de  $b$  vérifiant  $a \cdot |C_1| + b \cdot |C_2| = 0$ , e.g.  $a = (|C_1| + |C_2|) / |C_1|$  et  $b = -(|C_1| + |C_2|) / |C_2|$ . La solution qui minimise l'erreur carrée moyenne  $\|Y - W^t \cdot X\|^2$  est, [Dud73] :

$$W = \alpha \cdot \text{Var}^{-1} \cdot (\bar{x}_1 - \bar{x}_2) \quad (7.8)$$

Où  $\bar{x}_k$  représente la moyenne de la classe  $C_k$ ,  $\text{Var}$  est la matrice de covariance des données,  $\alpha$  est une constante et  $\bar{x}$  est la moyenne de tous les exemples. La fonction discriminante correspondante est :

$$g_R(x) = W^t \cdot (x - \bar{x}) = \alpha \cdot (\bar{x}_1 - \bar{x}_2)^t \cdot \text{Var}^{-1} \cdot (x - \bar{x}) = \alpha \cdot (\bar{x}_1 - \bar{x}_2)^t \cdot \text{Var}^{-1} \cdot x + A \quad (7.9)$$

où  $A = -\alpha \cdot (\bar{x}_1 - \bar{x}_2)^t \cdot \text{Var}^{-1} \cdot \bar{x}$



Et la règle de décision est décider  $C_1$  si  $g_R(x) > 0$  et  $C_2$  sinon.

En remplaçant la moyenne et la matrice de covariance dans (7.7) par leurs valeurs estimées obtenues sur les données, comme dans (7.9), les deux règles de décisions  $g_B(x)$  et  $g_R(x)$  sont équivalentes à un seuil près. Le seuil optimal estimé par la règle de décision bayésienne peut se calculer facilement à partir des données si on le désire. Il est donc possible à partir d'une approche de régression d'estimer la frontière de décision bayésienne de façon optimale au sens du maximum de vraisemblance sous les hypothèses de populations normales et d'égalité des matrices de covariance. Ceci dit, pour des applications pratiques, il n'y a aucune garantie que la règle de décision bayésienne - i.e. l'emploi de seuil "optimum" donne de meilleurs résultats qu'une approche de régression pure (cf. tableau 9).

### 7.3.2.3 Discrimination logistique

La discrimination logistique, a été introduite par les statisticiens vers la fin des années soixante [Cox66, Tru67] et popularisée principalement par Anderson [And72, And79, And82].

Il s'agit d'une tentative pour s'affranchir des hypothèses restrictives souvent associées aux méthodes linéaires paramétriques (comme les hypothèses de normalité et d'égalité des matrices de covariance utilisées en 7.3.2.1).

La seule hypothèse qui est faite est que le logarithme des rapports de probabilité conditionnelles des classes soit linéaire en  $x$ , ce qui dans le cas d'un problème de classification à deux classes donne:

$$\log\left(\frac{p(x/C_1)}{p(x/C_2)}\right) = \beta_0 + \beta^t \cdot x \quad (7.10)$$

$$\text{où } \beta^t = (\beta_1, \dots, \beta_p)$$

L'intérêt du modèle (7.10) est que les probabilités a posteriori ont alors une forme logistique simple :

$$\begin{aligned} p(C_1/x) &= \frac{\pi_1 \cdot p(x/C_1)}{\pi_1 \cdot p(x/C_1) + \pi_2 \cdot p(x/C_2)} = \frac{\frac{\pi_1}{\pi_2} \cdot \frac{p(x/C_1)}{p(x/C_2)}}{1 + \frac{\pi_1}{\pi_2} \cdot \frac{p(x/C_1)}{p(x/C_2)}} \\ &= \frac{\exp(\beta_0 + \log \frac{\pi_1}{\pi_2} + \beta^t \cdot x)}{1 + \exp(\beta_0 + \log \frac{\pi_1}{\pi_2} + \beta^t \cdot x)} \end{aligned} \quad (7.11)$$

$$p(C_2/x) = 1 - p(C_1/x)$$

L'utilité de cette méthode est qu'elle recouvre une grande variété de familles de distributions, par exemple toutes les distributions de la famille exponentielle [And75]. Les paramètres  $\beta_0$ ,  $\beta$  et  $K = \pi_1/\pi_2$  sont estimés grâce à une méthode numérique de façon à maximiser la vraisemblance des données [And82, Sap90] :

$$V = \prod_{x \in C_1} p(x/C_1) \prod_{x \in C_2} p(x/C_2)$$

Une fois ces paramètres estimés, la règle de décision dépendra seulement de la fonction linéaire :

$$g_L(x) = \beta^t \cdot x + \beta_0 + \log K \quad (7.12)$$

Cette règle consiste à décider  $C_1$  si  $g_L(x) > 0$  et  $C_2$  sinon.

Anderson étend cette idée au cas de l'apprentissage semi-supervisé [And79, And82]. Les exemples non étiquetés sont supposés provenir d'un mélange de densités et les exemples étiquetés de chacune des composantes [And79, And82]. Il propose de déterminer les paramètres du modèle en maximisant la vraisemblance de l'ensemble des données, qui s'écrit alors sous la forme :

$$V = \prod_{i=1}^{n+m} \{p(x_i / C_1)\}^{t_{1i}} \{p(x_i / C_2)\}^{t_{2i}} \{\pi_1 \cdot p(x_i / C_1) + \pi_2 \cdot p(x_i / C_2)\}^{t_{3i}} \quad (7.13)$$

Où si  $x_i \in C_k$  alors  $t_{ki} = 1$  et  $t_{(1-k)i} = 0$  pour  $k = \{1,2\}$ , et dans le cas où l'origine de  $x_i$  est inconnue alors  $t_{3i} = 1$  et  $t_{hi} = 0$  pour  $h \neq 3$ .

### 7.3.3 Modèles discriminants et apprentissage non-supervisé et semi-supervisé

#### 7.3.3.1 Algorithme CEM-regression

Nous allons montrer dans cette section que sous l'hypothèse de populations gaussiennes avec des matrices de covariances communes l'algorithme 18, pour un classifieur linéaire, est une instance de l'algorithme CEM dans le cas non-supervisé ou de l'algorithme CEM-semi-supervisé dans le cas d'apprentissage semi-supervisé. Pour simplifier la présentation, nous ne considérons que le cas non-supervisé. La démonstration est similaire pour le semi-supervisé.

Une différence majeure par rapport à CEM est que l'on utilise un modèle discriminant (une régression des entrées par rapport aux sorties désirées) au lieu d'un modèle génératif, d'où le nom *CEM-regression*.

En CEM, à la  $j^{\text{ème}}$  itération, les paramètres  $(\pi^{(j)}, \theta^{(j)})$  sont utilisés pour estimer les probabilités a posteriori de façon à assigner à l'étape  $C$ , les données aux différents groupes. Au lieu d'estimer ces probabilités à l'aide d'un modèle génératif, CEM-regression les estime directement à partir d'un modèle de discrimination linéaire.

Pour une partition  $P^{(j)}$  obtenue à la  $j^{\text{ème}}$  itération de l'algorithme CEM, supposons que nous faisons une régression de la matrice  $X$  sur le vecteur désiré  $Y$  (cf. section 7.3.2.2).

Dans ce cas, avec un seuil approprié, la règle de décision inférée par l'estimation de la régression correspondra à la règle de décision bayésienne optimale, où les paramètres des gaussiennes sont remplacés par les estimés au sens du maximum de vraisemblance.

En utilisant la fonction de décision du modèle (eq. 7.9) dont on change le terme constant  $A$  pour qu'il soit un estimateur du seuil optimal de Bayes (eq. 7.7), et en utilisant cette règle pour affecter aux données une des 2 classes (l'étape  $C$ ), la partition  $P^{(j+1)}$  sera

exactement la même que celle obtenue par la version estimation de densité de l'algorithme CEM.

De cette façon l'étape **E** de l'algorithme CEM peut être remplacée par une étape de régression et la décision prise à l'étape **C** sera inchangée.

Comme nous sommes intéressé uniquement par l'affectation des données à un groupe ou une classe, l'étape **M** n'est plus nécessaire dans notre approche. L'algorithme CEM-régression débute par une partition initiale, la  $j^{\text{ème}}$  étape consiste à classer les données non étiquetées grâce à la fonction de décision calculée. On peut démontrer facilement que l'algorithme CEM-régression converge vers un minimum local de la fonction de vraisemblance pour l'apprentissage semi-supervisé ou non-supervisé. Cet algorithme peut être décrit comme suit dans le cas non supervisé:

---

**Initialisation:** Commencer par une partition  $P^{(0)}$

Pour  $j = 0$  jusqu'à la convergence faire

- Etape – **E** : Estimer  $Var^{(j)}$ , et  $\bar{x}_k^{(j)}$  sur  $P_k^{(j)}$ , pour  $k \in \{1,2\}$

$$\text{Calculer } W^{(j)} = \alpha \cdot Var^{(j)^{-1}} \cdot (\bar{x}_1^{(j)} - \bar{x}_2^{(j)})$$

- Etape – **C** : Classer  $x_i$  suivant le signe de  $(W^{(j)}x_i + w_0^{(j)})$  en deux classes  $P_1^{(j+1)}$  ou  $P_2^{(j+1)}$ .

---

Algorithme 20. L'algorithme CEM-régression en non supervisé

Pour des problèmes plus complexes ou peut être moins bruités, des modèles plus efficaces comme des réseaux de neurones ou des MVS non linéaires peuvent être utilisés à la place d'un classifieur linéaire.

En semi-supervisé, comme dans l'algorithme 19, les données étiquetées sont utilisées pour définir une partition initiale, les étiquettes des données de  $D_l$  restent inchangées au cours de l'algorithme.

**Implémentation**

Cet algorithme s'implémente très simplement en utilisant un neurone linéaire et pour chaque itération un critère d'optimisation classique qui est l'erreur au sens des moindres carrés entre sorties désirées et calculées courantes.

L'étape E consiste à estimer les paramètres du regressueur par rapport aux sorties désirées courantes, ceci peut être fait algébriquement ou par une technique numérique de type gradient. L'étape C consiste à prendre une décision par rapport au regressueur courant. On retrouvera une implémentation similaire dans le cas logistique.

### 7.3.3.2 Algorithme CEM-logistique

Nous analysons l'algorithme 18 sous des hypothèses plus faibles qui sont celles de la discrimination logistique. Dans ce cadre, le logarithme du rapport des probabilités conditionnelles des classes pertinentes et non-pertinentes est une fonction linéaire des entrées.

Nous montrons qu'il s'agit là aussi d'une instance de l'algorithme CEM et que cet algorithme peut être implémenté très simplement de façon similaire au cas linéaire, en remplaçant le neurone linéaire par un neurone sigmoïde. Les deux implémentations sont donc quasiment similaires. Le résultat que nous démontrons est un peu plus général que celui prouvé précédemment.

Dans un premier temps nous montrons la convergence, puis nous considérons l'implémentation avec un modèle sigmoïde.

#### *Description de l'algorithme*

Nous considérons uniquement le cas non-supervisé, le cas semi-supervisé se traite de façon analogue. Nous partons de la vraisemblance de classification qui dans le cas à deux classes s'écrit (cf eq 7.5) :

$$L_C = \sum_{i=1}^m \{t_{1i} \cdot \log(\pi_1 \cdot p(x_i / P_1; \beta, \beta_0)) + t_{2i} \cdot \log(\pi_2 \cdot p(x_i / P_2; \beta, \beta_0))\} \quad (7.14)$$

Ce qui d'après la règle de Bayes donne:

$$L_C = \tilde{L}_C + \sum_{i=1}^m \log(p(x_i)) \quad (7.15)$$

$$\text{Où } \tilde{L}_C = \sum_{i=1}^m \{t_{1i} \cdot \log(p(P_1 / x_i; \beta, \beta_0)) + t_{2i} \cdot \log(p(P_2 / x_i; \beta, \beta_0))\}$$

On cherche l'optimum de  $L_C$  par rapport aux paramètres  $\beta$  et  $t$ . Comme aucune hypothèse n'a été faite sur la forme fonctionnelle de  $p(x)$ , l'estimateur du maximum de vraisemblance de  $\beta$  pour  $L_C$  est le même que celui qui maximise  $\tilde{L}_C$  [McL92, p.261, AND82].

Pour maximiser  $\tilde{L}_C$  il n'existe pas de méthode analytique, nous utiliserons donc pour cela une méthode numérique, en l'occurrence la méthode de quasi-Newton [Gill72] ; l'avantage de cette méthode est qu'à chaque itération elle n'a besoin que des dérivées d'ordre premier pour le calcul du maximum. En posant  $x_0=1$ , il vient d'après la forme logistique des probabilités a posteriori que :

$$\frac{\partial p(P_1 / x)}{\partial \beta_j} = x_j \cdot p(P_1 / x) \cdot (1 - p(P_1 / x)), \quad \forall j \in \{0, \dots, p\}$$

$$\frac{\partial p(P_2 / x)}{\partial \beta_j} = -\frac{\partial p(P_1 / x)}{\partial \beta_j}, \quad \forall j \in \{0, \dots, p\} \quad (7.16)$$

en substituant ces dérivées dans (7.15) il vient alors:

$$\frac{\partial \tilde{L}_C(P, K, \beta, \beta_0)}{\partial \beta_j} = \sum_{i=1}^m \{t_{1i} - p(P_1 / x_i; \beta, \beta_0)\} x_{ij}, \quad \forall j \in \{0, \dots, p\} \quad (7.17)$$

L'optimisation des paramètres  $t$ , se fait dans l'algorithme CEM à l'étape C.

L'algorithme CEM-logistique dans le cas non-supervisé peut alors s'écrire (algorithme21)

---

**Initialisation:** des paramètres,  $\beta_0^{(0)}, \dots, \beta_p^{(0)}, t_{ki}^{(0)} i = 1..m, k = 1,2$ , par une régression logistique classique, en utilisant une partition initiale  $P^{(0)}$  obtenue dans le cas du résumé par l'algorithme de base.

$$K^{(0)} = \frac{\sum_{i=1}^m t_{1i}^{(0)}}{\sum_{i=1}^m t_{2i}^{(0)}} \quad (\text{estimation du rapport de proportion des 2 classes})$$

Pour  $j \geq 0$ , faire jusqu'à convergence

- Etape – **E** : Calculer les probabilités a posteriori :

$$p(P_1^{(j)} / x; K^{(j)}, \beta^{(j)}, \beta_0^{(j)}) = \frac{\exp(\beta_0^{(j)} + \log K^{(j)} + \beta^{(j)} \cdot x)}{1 + \exp(\beta_0^{(j)} + \log K^{(j)} + \beta^{(j)} \cdot x)}$$

$$p(P_2^{(j)} / x; K^{(j)}, \beta^{(j)}, \beta_0^{(j)}) = 1 - p(P_1^{(j)} / x; K^{(j)}, \beta^{(j)}, \beta_0^{(j)})$$

- Etape – **C** : Déterminer la classification optimale (au sens de Bayes)

$$\forall i, \forall k \in \{1,2\}, t_{ki}^{(j+1)} = \text{sgn}(p(P_k^{(j)} / x_i; K^{(j)}, \beta^{(j)}, \beta_0^{(j)}) - 0.5)$$

L'affectation des étiquettes revient à prendre la classe donnant la plus grande probabilité a posteriori pour l'exemple.

$$K^{(j+1)} = \frac{\sum_{i=1}^m t_{1i}^{(j+1)}}{\sum_{i=1}^m t_{2i}^{(j+1)}}$$

- Etape – **M** : Estimer les paramètres  $\beta_0^{(j+1)}$  et  $\beta^{(j+1)}$  maximisant  $L_C$ :

$$\tilde{L}_C(P^{(j+1)}, K^{(j+1)}, \beta^{(j)}, \beta_0^{(j)}) = \sum_{i=1}^m \sum_{k=1}^2 \{t_{ki}^{(j+1)} \cdot \log(p(P_k^{(j+1)} / x_i; K^{(j+1)}, \beta^{(j)}, \beta_0^{(j)}))\}$$

suivant une méthode numérique (quasi-Newton dans notre cas).

---

Algorithme 21. L'algorithme CEM-logistique - cas non-supervisé

### Convergence de l'algorithme

La convergence de cet algorithme s'établit simplement en remarquant que :

- A l'étape **C** on a choisit les classes suivant la règle optimale de classification de Bayes soit :

$$\tilde{L}_C(P^{(j+1)}, K^{(j+1)}, \beta^{(j)}, \beta_0^{(j)}) \geq \tilde{L}_C(P^{(j)}, K^{(j)}, \beta^{(j)}, \beta_0^{(j)}) \quad (7.18)$$

- A l'étape **M**, on cherche de nouveaux paramètres  $\{\beta_0^{(j+1)}, \beta^{(j+1)}\}$  qui maximisent  $\tilde{L}_C(P^{(j+1)}, K^{(j+1)}, \beta^{(j)}, \beta_0^{(j)})$ , soit :

$$\tilde{L}_C(P^{(j+1)}, K^{(j+1)}, \beta^{(j+1)}, \beta_0^{(j+1)}) \geq \tilde{L}_C(P^{(j+1)}, K^{(j+1)}, \beta^{(j)}, \beta_0^{(j)}) \quad (7.19)$$

Soit d'après (7.15) et (7.16) à chaque itération, on a :

$$\tilde{L}_C(P^{(j+1)}, K^{(j+1)}, \beta^{(j+1)}, \beta_0^{(j+1)}) \geq \tilde{L}_C(P^{(j)}, K^{(j)}, \beta^{(j)}, \beta_0^{(j)}) \quad (7.20)$$

### Implementation

Nous avons implémenté cet algorithme avec un neurone qui possède une fonction d'activation sigmoïde de pente 1 à l'origine :

$$F: \mathbb{R} \rightarrow [0,1]$$

$$x \mapsto \frac{1}{1 + e^{-x}}$$

L'activation de la sortie est obtenue par une composition :

- d'un produit scalaire  $a$  des  $p$  entrées  $\{x_i\}_{i=1..p}$  et de l'entrée biais  $x_0=1$  avec les poids synaptiques  $\{\beta_i\}_{i=1..p}$  reliant les entrées à la sortie et le paramètre du biais  $\beta_0$  :

$$a = \sum_{i=1}^p \beta_i \cdot x_i + \beta_0 = \beta^t \cdot x + \beta_0$$

- et de la fonction de transfert  $F(\cdot)$  :

$$y = F(a)$$

On note que pour chaque exemple  $x$  la sortie correspondante  $y$  est une estimation de la probabilité d'appartenance à la partition  $P_1$  :

$$y = \frac{1}{1 + \exp(-(\beta_0 + \beta^t \cdot x))} = \frac{\exp(\beta_0 + \beta^t \cdot x)}{1 + \exp(\beta_0 + \beta^t \cdot x)} = p(P_1 / x)$$

L'autre probabilité étant obtenue par  $p(P_2/x) = 1 - p(P_1/x)$ . Le critère d'optimisation que l'on prend est le logarithme de la vraisemblance de classification  $\tilde{L}_C$ . On cherche donc à apprendre les poids du modèle qui maximise ce critère. Nous remarquons que les dérivées partielles de ce critère sont de la forme :

$$\tilde{L}_C = \sum_{i=1}^m E_i, \quad (7.21)$$

$$\text{où } E_i = t_{li} \log p(P_1 / x_i, \beta, \beta_0) + (1 - t_{li}) \log(1 - p(P_1 / x_i, \beta, \beta_0)).$$

On peut en déduire les dérivées partielles de  $\tilde{L}_C$  en fonction de celles de  $E_i$  soit d'après (7.16) :

$$\frac{\partial L_C}{\partial \beta_j} = \sum_{i=1}^m \frac{\partial E_i}{\partial \beta_j}, \quad (7.22)$$

$$\text{avec } \frac{\partial E_i}{\partial \beta_j} = (t_{li} - y_i) \cdot x_{ij}.$$

Ces dérivées sont directement utilisées par l'algorithme de gradient à l'étape M.

On peut implémenter la discrimination logistique avec un neurone sigmoïde, en utilisant pour déterminer les paramètres  $\beta$ , un algorithme de gradient (quasi-newton dans notre cas) pour maximiser la vraisemblance de classification.

On retrouve ainsi une implémentation classiquement utilisée dans le domaine des réseaux de neurones.

Le schéma d'apprentissage d'après cet algorithme est le suivant (figure 45) :

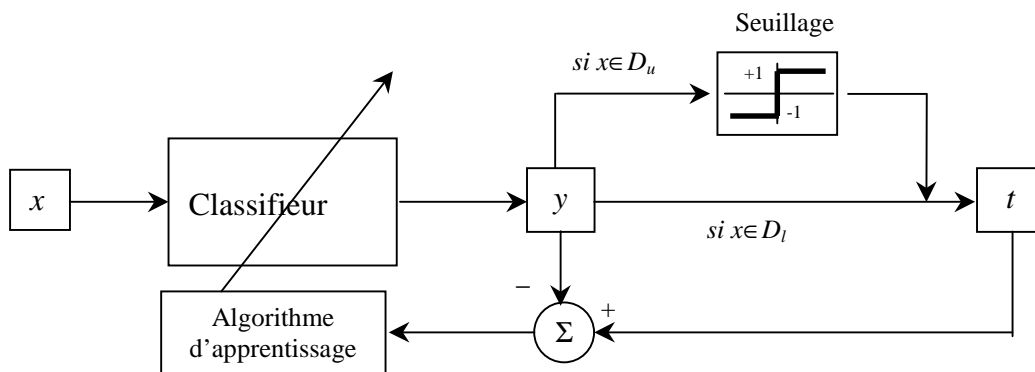


Figure 45. Modèle d'apprentissage basé sur l'algorithme CEM-logistique.

Les 2 algorithmes CEM-régression et CEM-logistique sont deux instances de l'algorithme CEM. Ils maximisent la vraisemblance de classification ou la forme modifiée (eq. 7.21) de cette vraisemblance pour le cas semi-supervisé. Ils peuvent être implémentés par des modèles de régression très simples qui sont d'une part un modèle linéaire et d'autre part un modèle logistique. D'un point de vue algorithmique, les deux implémentations sont quasi identiques. Ces modèles sont bien plus simples que les modèles génératifs proposés dans la littérature pour faire du semi-supervisé. Ils peuvent être étendus de façon très naturelle en utilisant d'autres types de classifieurs.

## 7.4 Base de Données utilisée pour les expériences

Un corpus de documents avec les résumés correspondants est nécessaire pour l'évaluation des systèmes. Nous avons utilisé pour cela la collection Reuters [Reuters], constituée de 1000 dépêches et de leur résumés associés. Les résumés des dépêches sont constitués de phrases extraites de ces dernières.

Base de donnée Reuters			
Collection	Apprentissage	Test	Total
# de documents	300	700	1000
Moyenne # de phrases /doc	26.18	22.29	23.46
Min phrase/doc	7	5	5
Max phrase/doc	87	88	88
Résumés			
Moyenne # de phrase /résumé	4.94	4.01	4.3
% de résumés contenant la première phrase des docs	63.3	73.5	70.6

Tableau 6. Caractéristiques de la base de données Reuters et des résumés associés

Cette base de données est divisée en une base d'apprentissage et une base de test. Les résumés de la base correspondent à des résumés génériques. Les systèmes que nous avons décrits nécessitent une requête pour calculer des similarités ou des scores pour simuler une requête adéquate pour ce type de résumé. Nous avons utilisé l'ensemble des mots les plus fréquents de la base d'apprentissage pour cela. Les statistiques sur les documents et leur résumé associé sont décrits dans le tableau 6.

La figure 46-gauche, montre l'histogramme de la longueur des résumés en nombre de phrases, la moyenne des résumés est d'environ 5 phrases. La figure 46-droite montre l'histogramme de la longueur des résumés en nombre de mots qui est approximativement une distribution normale avec un pic autour de 80 mots.

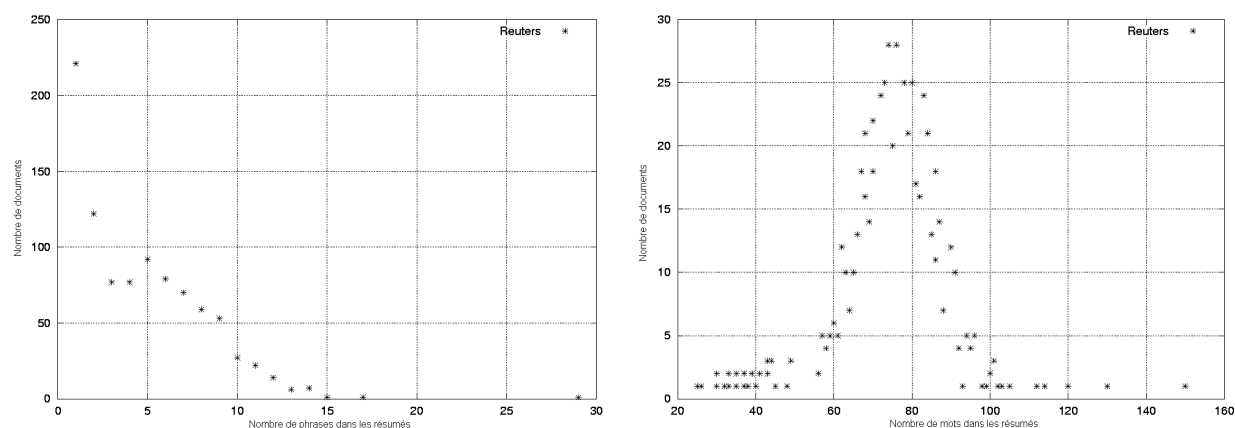


Figure 46. Gauche. Distribution de la longueur des résumés en nombre de phrases, droite. Distribution de la longueur des résumés en nombre de mots.



## 7.5 Stratégie d'apprentissage

Nous allons mener deux séries d'évaluations :

1. La première pour tester l'efficacité de l'apprentissage semi-supervisé et non-supervisé
2. Le deuxième pour tester nos deux algorithmes de discrimination : CEM-régression et CEM-logistique.

### 7.5.1 Espace de représentation

Afin d'entraîner les deux systèmes pour la tâche de classification de phrases, nous considérons chaque phrase comme une séquence de mots où les termes sont représentés par un vecteur.

Pour cela, nous utilisons 4 valeurs pour représenter chaque terme  $w$  d'une phrase  $\varphi$  :  $tf(w, \varphi)$ ,  $\bar{tf}(w, q)$ ,  $(1 - (\log(df(w)+1)/\log(n+1)))$  et  $Sim_2(q, \varphi)$  la similarité entre la phrase  $\varphi$  et la requête  $q$  – ces quantités sont calculées comme expliqué dans la section 7.2.1.

Les trois premières valeurs sont des statistiques de fréquences qui donnent l'importance d'un terme pour caractériser respectivement la phrase, la requête et le document. La dernière composante donne l'importance de la phrase  $\varphi$  contenant  $w$  pour le résumé. Cette similarité est calculée pour la phrase qui contient  $w$  plutôt que sur  $w$  car il est difficile de mesurer des similarités à partir de termes isolés [Kna94].

### 7.5.2 Apprentissage interactif

Dans le cas de l'apprentissage interactif, et afin de fournir un premier étiquetage des phrases pour entraîner notre classifieur, nous avons procédé en deux étapes :

1. On utilise le système de base décrit en (7.2.1), puis on fixe empiriquement et de façon automatique un seuil sur les mesures de similarités obtenues sur les phrases pour un document donné, les phrases ayant une mesure plus grande que ce seuil sont étiquetées comme pertinentes et les autres comme non-pertinentes
2. L'utilisateur peut changer l'étiquetage de ces phrases.

Le classifieur est ensuite entraîné pour apprendre  $p(P_q/\varphi)$  en utilisant ces informations.

### 7.5.3 Apprentissage semi-supervisé et non-supervisé avec les algorithmes CEM-régression et CEM-logistique

Dans le cas de l'apprentissage semi-supervisé, nous avons étiqueté 10% des phrases de la base d'apprentissage en utilisant les résumés associés aux documents comme étant l'ensemble des phrases étiquetées pertinentes. Les étiquettes de ces phrases sont gardées inchangées durant toute la phase d'apprentissage.

Pour l'apprentissage non-supervisé, nous avons obtenu un premier étiquetage des phrases grâce au système de base qui utilisée une mesure de similarité entre requête et phrases.

Avec les algorithmes CEM-régression et CEM-logistique, un premier classifieur est entraîné en utilisant les étiquettes initiales fournies soit dans l'ensemble d'apprentissage (cas semi-supervisé) soit par l'algorithme de base (cas non supervisé) - c'est l'étape d'initialisation des algorithmes. Puis on entre dans les itérations CEM. A partir de la décision de ce classifieur initial, on en déduit les nouvelles étiquettes des phrases de la base d'apprentissage non étiquetée. Ces nouvelles étiquettes sont ensuite réutilisées pour entraîner un nouveau classifieur et ainsi de suite jusqu'à la convergence.

## 7.6 Evaluation

Comme nous l'avons mentionné au chapitre 5, l'évaluation des systèmes de résumé a fait l'objet de plusieurs tentatives, dont la plupart ont été réalisées pour les campagnes Tipster [NIST93] et Summac [SUM98].

Ce problème reste difficile ; différents aspects doivent être considérés simultanément afin de pouvoir évaluer et comparer les systèmes de résumés [Mitt99].

Nos méthodes fournissent un ensemble de phrases pertinentes (autant qu'il y en a dans les résumés associés). En prenant toutes les phrases sélectionnées, nous pouvons construire un *extrait* pour le document. Pour l'évaluation nous avons comparé cet extrait avec les phrases du résumé associé au document dans la base test.

L'évaluation est alors faite grâce aux mesures de précision et de rappel définies comme suit :

$$\text{Précision} = \frac{\# \text{ nombre de phrases extraites par le système qui sont dans les résumés associés}}{\# \text{ total de phrases extraites par le système}}$$

$$\text{Rappel} = \frac{\# \text{ nombre de phrases extraites par le système qui sont dans les résumés associés}}{\# \text{ total de phrases dans les résumés}}$$

### 7.6.1 Apport de l'interaction

Pour quantifier l'importance de l'effet de l'interaction, nous avons fait plusieurs test en variant le degré d'interaction. Pour simuler l'interaction utilisateur nous avons considéré un sous-ensemble  $\Delta$  (sous-ensemble d'interaction) de l'ensemble d'apprentissage. Les étapes de cette procédure itérative sont les suivantes :

1. En se basant sur les décisions du système de base (cf. section 7.2.1), on apprend le classifieur en mode non-supervisé,
2. Nous allons ensuite utiliser les résumés associés aux documents du sous-ensemble d'interaction pour changer les étiquettes des phrases de ces documents (utiliser les vraies étiquettes des phrases) pour apprendre à nouveau le classifieur. Pour les phrases des documents n'apparaissant pas dans le sous-ensemble d'interaction on utilisera les étiquettes prédites par le classifieur.

La figure 47 illustre cette procédure. Notons que cette interaction simulée revient à conjuguer les deux types d'apprentissage non-supervisé et semi-supervisé.

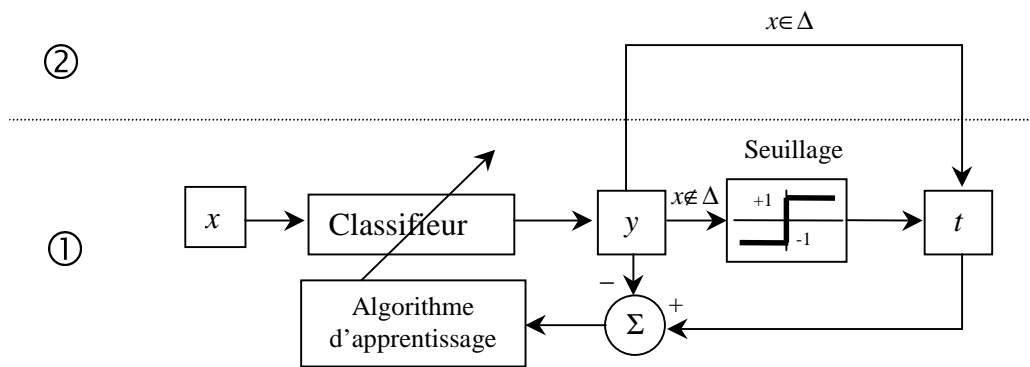


Figure 47. La simulation de l'interaction se fait en deux étapes. A l'étape ① le classifieur apprend en mode non-supervisé grâce à un premier étiquetage obtenu par le système de base. A l'étape ②, on change les étiquettes des phrases qui sont dans le sous-ensemble d'interaction.

La figure 48 montre le comportement de l'algorithme pour différentes tailles du sous-ensemble d'interaction. Les performances augmentent graduellement avec le degré de l'interaction.

Pour la comparaison nous avons aussi regardé les performances du système lorsque le classifieur est entraîné sans expansion de la requête. Les performances dans ce cas sont plus faibles et elles atteignent un plateau avant de décroître probablement à cause du sur-apprentissage.

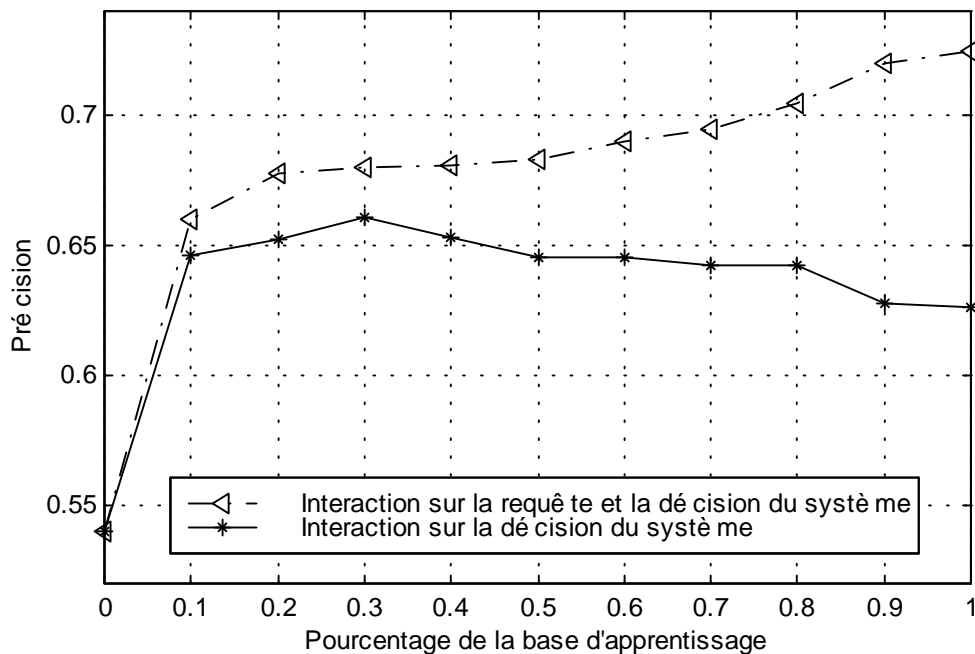


Figure 48. La précision du système en fonction du degré d'interaction. L'axe des abscisses représente le pourcentage de documents dans la base d'apprentissage utilisés pour l'interaction.

### 7.6.2 Apprentissage semi-supervisé et non-supervisé vs. Système de base

Dans cette section nous allons comparer le système de base avec le modèle d'apprentissage entraîné grâce à l'algorithme CEM-régression en mode semi-supervisé et non-supervisé.

Dans la section suivante nous allons aussi comparer les deux modèles CEM-régression et CEM-logistique. Nous verrons que ces deux modèles donnent à peu près les mêmes performances sur la base test.

Les phrases extraites par le système entraîné grâce à l'algorithme CEM-regression pour les différentes stratégies d'apprentissage sont comparées à celles des résumés de la base de test.

Nous avons obtenu 53,94% de précision et 56,33% de bonne classification<sup>14</sup> avec le système de base. Afin de donner une borne supérieure aux performances du classifieur nous avons utilisé tous les résumés associés aux documents de la base d'apprentissage pour avoir un étiquetage complet des phrases de cet ensemble et pouvoir donc faire de l'apprentissage complètement supervisé. En mode supervisé avec le classifieur linéaire, nous avons obtenu 72,68% de précision et 74,06% de rappel.

Nous avons obtenu des performances en mode non-supervisé et semi-supervisé qui sont toutes deux très similaires . Dans les deux cas, les mesures de précision et de performance totale augmentent d'à peu près 10 points par rapport au système de base.

	Précision (%)	Performance totale (%)
Système de base	54,94	56,33
Apprentissage non-supervisé	63,53	64,92
Apprentissage semi-supervisé	63,94	65,32
Apprentissage interactif	66,03	66,89
Apprentissage supervisé	72,68	74,06

Tableau 7. Comparaison entre le système de base et les différentes techniques d'apprentissage avec l'algorithme CEM-régression utilisant un neurone linéaire comme classifieur. Les performances sont sur la base test.

Pour l'apprentissage interactif, lorsque nous utilisons 10% de la base d'apprentissage pour l'interaction, (ce qui revient à coupler les deux types d'apprentissage non-supervisé et semi-supervisé) les performances sont augmentées de 3% par rapport à la stratégie de résumé automatique précédente. Le tableau 7, récapitule les différentes performances obtenues.

<sup>14</sup> On définit :

$$\text{La précision par Précision} = \frac{\text{\#nombre de phrases extraites par le système qui sont dans les résumés associés}}{\text{\# total de phrases extraites par le système}}$$

$$\text{Le taux de bonne classification par PBC} = \frac{\text{\#de phrases bien classées}}{\text{\#total de phrases}}$$

	Précision (%)	Performance totale (%)
Apprentissage Semi-supervisé avec un modèle linéaire	63,94	65,32
Apprentissage semi-supervisé avec un MVS	62,45	63,75

Tableau 8. Comparaison entre deux modèles : linéaires et Machine à Vecteurs Supports dans le cas de l'apprentissage semi-supervisé avec l'algorithme *CEM-régression*. Les performances sont sur la base de test.

Si nous comparons les résultats du classifieur en mode non-supervisé et semi-supervisé avec le mode supervisé nous noterons qu'elles permettent d'augmenter les performances du classifieur de base de 10 points environ par rapport à 20 points pour le mode supervisé. Dans notre exemple, ces méthodes sont capables d'extraire des données non étiquetées, la moitié de l'information utilisée par une classification supervisée.

Nous avons aussi comparé le classifieur *CEM-régression* avec une Machine à Vecteurs Supports à noyau linéaire (tableau 8). Les modèles ont des performances similaires.

Nous nous sommes aussi intéressé à l'équivalence entre les deux règles de décisions  $g_B(x)$  (7.7) et  $g_R(x)$  (7.9). Comme la différence entre ces deux fonctions réside seulement dans le terme constant qui dans le cas  $g_B$  mesure la proportion d'individus dans les deux classes à discriminer, il était bon de connaître l'importance de ce seuil.

Le résultat de la comparaison entre les deux règles (7.7) et (7.9) est résumée dans le tableau 9 (pour les deux stratégies d'apprentissage non-supervisé et semi-supervisé). Nous notons que la précision de la classe pertinente décroît légèrement tandis que la performance globale augmente. Les performances sont très similaires.

	Précision (%)	Performance totale (%)
Classification avec le seuil optimal bayésien – cas semi-supervisé	63,15	65,45
Classification sans le seuil optimal bayésien – cas semi-supervisé	63,94	65,32
Classification avec le seuil optimal bayésien – cas non-supervisé	62,98	65,27
Classification sans le seuil optimal bayésien – cas non-supervisé	63,53	64,92

Tableau 9. Comparaison entre les fonctions de discrimination (7.7) et (7.9) dans les deux cas d'apprentissage semi-supervisé et non-supervisé.

Les courbes de rappel/précision à 11 points permettent une évaluation plus précise du comportement du système (figure 49). Posons  $M$ , le nombre total des phrases pertinentes extraites par le système (d'une manière correcte ou incorrecte),  $N\phi$  le nombre total de phrases extraites par le système qui sont dans les résumés,  $N_g$  le nombre total de phrases dans les résumés, et  $N_t$  le nombre total de phrases dans la base test.

La précision et le rappel sont calculés respectivement comme les rapports  $N\phi/M$  et  $N\phi/N_g$ . Pour un document donné les phrases sont triées suivant les décisions du système. Les mesures rappel et précision sont alors calculées pour  $M=1,\dots,N_t$  et sont ensuite interpolées sur 11 points.

Les courbes révèlent les mêmes comportements que le tableau 7, l'apprentissage semi-supervisé et non-supervisé ont des performances similaires et pour toutes les valeurs de rappel l'augmentation de performance est la moitié de celle obtenue avec le système supervisé. L'apprentissage non-supervisé est donc prometteur du fait qu'il n'a besoin d'aucune interaction ni d'aucun étiquetage. Nous remarquons que cette méthode peut être implémentée de la même façon pour la tâche de résumé à base de requête.

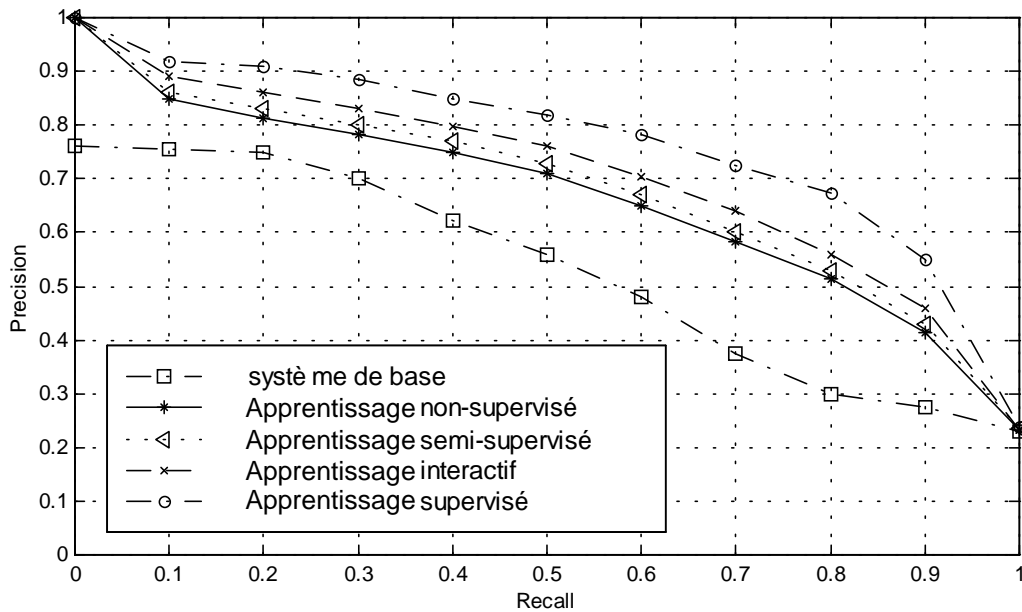
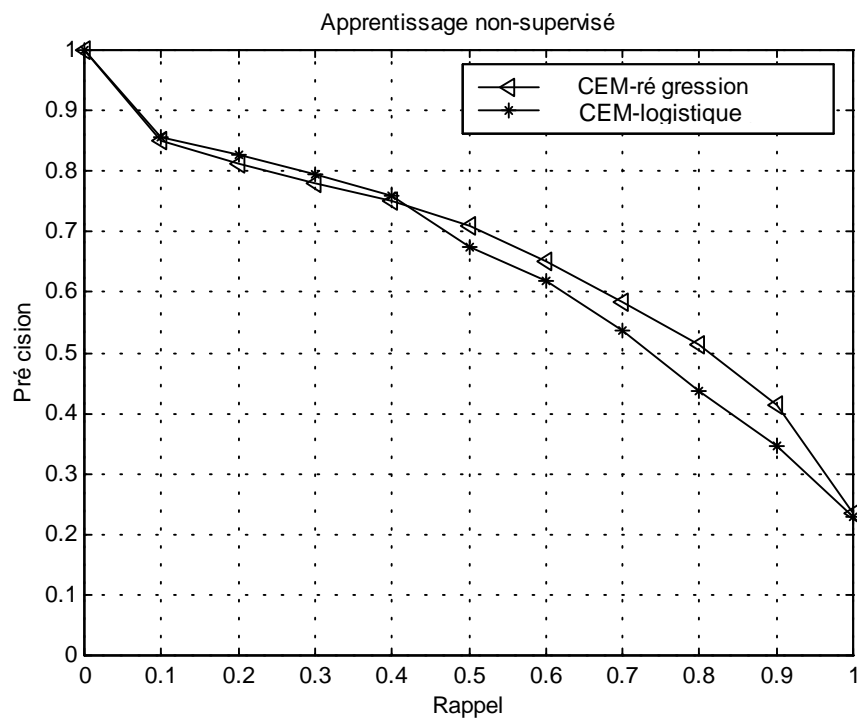


Figure 49. Les courbes de rappel et de précision pour le système de base (carré), apprentissage non-supervisé (étoile), apprentissage semi-supervisé (triangle), apprentissage interactif (croix) et apprentissage supervisé (cercle). Le classifieur utilisé est le classifieur avec un neurone sigmoïde.

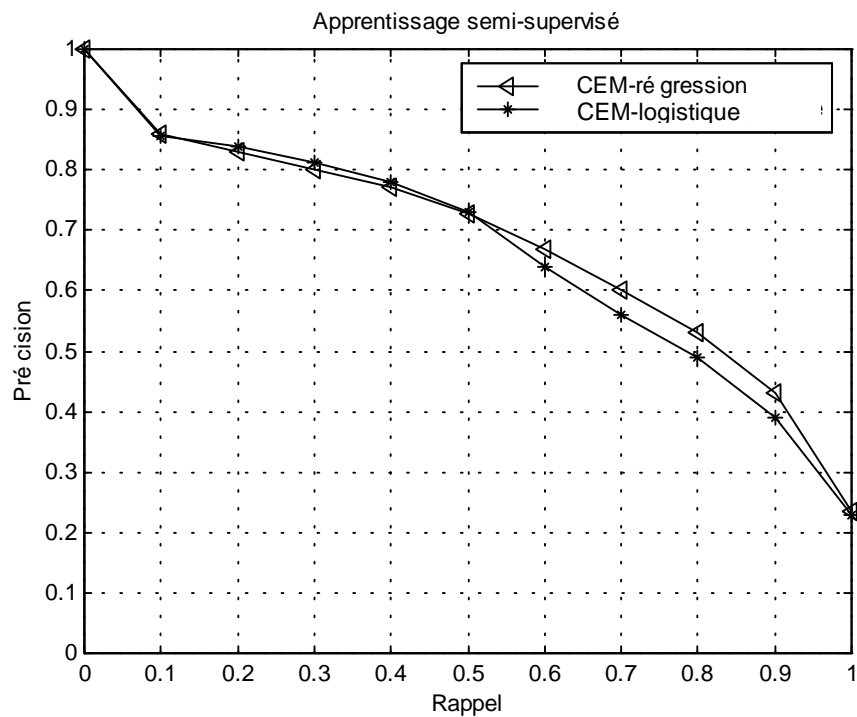
### 7.6.3 Comparaison des algorithmes CEM-régression et CEM-logistique

Nous avons aussi comparé les deux modèles CEM-régression et CEM-logistique. L'avantage du deuxième modèle est que l'hypothèse de base pose moins de contraintes que le modèle CEM-régression et qu'elle est vérifiée par beaucoup de distributions.

La figure 50 montre la comparaison des courbes rappel-précision suivant les deux schémas d'apprentissage non-supervisé (figure 50, a) et semi-supervisé (figure 50, b). Les deux modèles ont des comportements analogues. Toutefois nous notons que pour les deux stratégies d'apprentissage (semi-supervisé et non-supervisé) le modèle CEM-régression donne des résultats légèrement meilleurs pour des fortes valeurs de rappel et que ce phénomène est inversé pour le modèle CEM-logistique pour des faibles valeurs de rappel.



(a)



(b)

Figure 50. Comparaison des courbes de rappel et de précision entre l'algorithme de CEM-régression et la CEM-logistique dans le cas de l'apprentissage non-supervisé (a) et de l'apprentissage semi-supervisé (b).

## 7.7 Conclusion

Nous avons présenté deux techniques d'apprentissage pour le résumé automatique de texte, basés sur l'extraction des phrases pertinentes d'un document. Les systèmes proposés trient les phrases d'un document en regard de leur pertinence par rapport à une requête utilisateur.

La première technique est basée sur l'interaction et elle permet d'adapter le système aux besoins d'utilisateurs et au corpus. Avec cette technique, nous n'avons pas besoin d'une collection étiquetée de documents. Bien que ce système permette de faire abstraction de tout étiquetage il est lourd à manipuler.

La deuxième technique prend avantage d'un petit sous-ensemble de documents étiquetés et a des performances un peu plus faibles que celles obtenues par interaction, mais elle a l'avantage d'être totalement automatique. Cette deuxième technique a été implémentée par deux modèles basés sur l'algorithme CEM, appelés CEM-logistique et CEM-régression. Nous avons donné la preuve de convergence de ces deux modèles ainsi que leur implémentation par des modèles simples de l'apprentissage numérique. L'originalité de notre approche est que nos modèles adoptent une approche discriminante au lieu d'estimer des densités conditionnelles comme cela est le cas dans la plupart des méthodes basées sur des techniques non-supervisées et semi-supervisées. A notre connaissance, l'algorithme CEM n'avait pas, lui non plus, été employé dans le cadre discriminant. L'idée intuitive de ces modèles est simple et consiste à utiliser un classifieur qui apprend à partir de ses propres sorties.

Nous avons proposé des implémentations simples et légères à mettre en œuvre de ces algorithmes.

Les deux techniques ont été testées sur la base de documents + résumés de Reuters, elles permettent d'atteindre une augmentation de 10 points par rapport à un système de base, ce qui est à comparer avec 20 points d'augmentation pour une supervision complète. En ce qui concerne l'application résumé automatique, nous n'avons pas considéré dans cette étude les problèmes de taux de compression souhaités pour les documents. C'est un problème important dans le domaine et qui a suscité de nombreuses études. Toutefois, on peut mentionner les travaux de [Gol99] qui tire la conclusion que le rapport entre la taille d'un document et de son résumé (qui reflète l'idée globale du document) est une constante.

Nous n'avons pas, non plus, étudié le cas des phrases ayant un sens similaire (les doublons) qui peuvent être extraites par nos systèmes. Ce traitement est compliqué du fait qu'il faut considérer le sens de chaque phrase dans le résumé et il nécessite un traitement plus linguistique qui sort du cadre que nous nous étions fixé.

Il y a très peu d'études sur l'apprentissage semi-supervisé en particulier à partir de modèles discriminants. Notre travail constitue une avancée dans ce domaine, mais beaucoup de questions restent encore ouvertes. En particulier il reste à trouver un cadre formel pour mesurer l'apport des données non étiquetées, et les qualités respectives des techniques génératives et discriminantes restent à éclaircir.



Enfin, les techniques que nous avons proposées sont générales et en particulier dans le cadre de la RI, elles peuvent être utiles pour d'autres tâches que le résumé, par exemple pour la recherche ad-hoc ou le filtrage.

## 7.8 Bibliographie

- [Alb78] Albert A., Quelques apports nouveaux à l'analyse discriminante. *Thèse de Doctorat*, 1978, Faculté des Sciences, Université de Liège.
- [Ami00] Amini M.R. Interactive Learning for Text Summarization. *Proceedings of PKDD'2000/MLTIA'2000 Workshop on Machine Learning and Textual Information Access*, 2000, pp. 44-52, Lyon France.
- [Ami01a] Amini M.R., Gallinari P. Self-Supervised Learning for Automatic Text Summarization by Text-span Extraction. *Proceedings of 23<sup>rd</sup> BCS European Annual Colloquium on Information Retrieval*. 2001, pp. 55-63, Darmstadt, Allemagne.
- [Ami01b] Amini. M.R., Gallinari P. Automatic Text Summarization using Unsupervised and Semi-supervised learning. *Proceedings of 11<sup>th</sup> International Conference of Artificial Neural Networks*.
- [Ami01c] Amini. M.R., Gallinari P. Learning for Text Summarization using Labeled and Unlabeled Sentences. *12th European Conference on Machine Learning*, 2001, à paraître.
- [And72] Anderson J.A., Separate sample logistic discrimination. *Biometrika*. 1972, Vol. 59, pp.19-35.
- [And75] Anderson J.A., Quadratic logistic discrimination. *Biometrika*. 1975, Vol. 62, pp.149-154.
- [And79] Anderson J.A., Multivariate logistic compounds. *Biometrika*. 1979, Vol. 66, pp. 17-26.
- [And82] Anderson J.A., Logistic Discrimination. *Handbook of Statistics*. 1982, Vol. 2, pp. 169-191.
- [Cox66] Cox D.R., Some procedures associated with the logistic qualitative response curve. *Research Papers in Statistics: Festschrift for J. Neyman*, eds. David, 1966, pp. 51-71.
- [CEL92] Celeux G., Govaert G. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics & Data Analysis*. 14 (1992) 315-332.
- [CHU00] Chuang W.T., Yang J. Extracting sentence segments for text summarization: a machine learning approach. *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 2000, pp. 152--159, Athens, Greece.

- [Dud73] Duda R. O., Hart P. T. Pattern Recognition and Scene Analysis. Edn. Wiley (1973).
- [Flu97] B. Flury. A first Course in Multivariate Statistics. *Springer*, 1997.
- [Gil72] Gill P.E., Murray W., Quasi-Newton methods for unstructured optimisation. *Journal of Mathematics Applications*, Vol. 9, pp. 91-108.
- [Gol99] Goldstein J., Kantrowitz M., Mittal V., Carbonell J. Summarizing Text Documents: Sentence Selection and Evaluation Metrics. *Proceedings of SIGIR'99*, 1999.
- [Jin99] Jing H., McKeown K. The decomposition of Human-Written Summary Sentences. *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99)*, 1999, CA.
- [Kna94] Knaus D., Mittendorf E., Schauble P., Sheridan P. Highlighting Relevant Passages for Users of the Interactive SPIDER Retrieval System. *TREC-4 proceedings*.
- [KUP95] Kupiec J., Pedersen J., Chen F. A Trainable Document Summarizer. *Proceedings of the 18th ACM SIGIR conference on research and development in information retrieval*, 1995, pp. 68-73, Seattle, USA.
- [Man98] Mani I., Bloedorn E. Machine Learning of Generic and User-Focused Summarization. *Proceedings of the Fifteenth National Conference on AI*, 1998, pp. 821-826.
- [McL92] McLachlan G.J.: Discriminant Analysis and Statistical Pattern Recognition. *Wiley*, New-York, 1992.
- [Mitt99] Mittal V., Kantrowitz M., Goldstein J., Carbonell J. Selecting Text Spans for Document Summaries: Heuristics and Metrics. *Proceedings of AAAI'99*, 1999, Orlando. USA.
- [NIST93] NIST . TIPSTER Information-Retrieval Text Research Collection, on CD-ROM. *Published by The National Institute of Standards and Technology*, Gaithersburg, Maryland, 1993.
- [POR80] M. F. Porter. An Algorithm for Suffixe Stripping. *Program*, Vol. 13, N° 3, pp. 130--137, 1980.
- [Reuters] <http://boardwatch.internet.com/mag/95/bwm9.html>

- [Roc 71] Rocchio J. Relevance feedback in Information Retrieval. *Smart Retrieval System: Experiments in Automatic Document Processing, Book*, Chapter 14, pp. 313-323, Prentice-Hall, 1971.
- [Sap90] G. Saporta, Probabilités Analyse des données et statistique. *Editions Technip*, 1990.
- [SUM98] TIPSTER Text Summarization Evaluation Conference (SUMMAC). [http://www-nlpir.nist.gov/related\\_projects/tipster\\_summac/](http://www-nlpir.nist.gov/related_projects/tipster_summac/).
- [Teu97] Teufel S., Moens M. Sentence Extraction as a Classification Task. *Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*. 1997, pp. 58-65, Madrid, Spain.
- [Tru97] Truett J., Cornfield J., Kannel W., A multivariate analysis of the risk of coronary heart disease in Framlington. *Journal of Chronic Diseases*. Vol. 20, pp.511-24.
- [WEB99] A. Webb. Statistical Pattern Recognition. *Oxford University Press Inc.*, 1999.
- [Xu96] Xu J., Croft W.B. Query Expansion Using Local and Global Document Analysis. *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 4--11, 1996.
- [Zec96] Zechner K. Fast Generation of Abstracts from General Domain Text Corpora by Extracting Relevant Sentences. *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pp. 986--989, Denmark.

## Discussion

Nous nous sommes intéressé dans cette thèse au développement de systèmes d'apprentissage automatique pour deux problèmes d'accès à l'information textuelle. Ces problèmes sont respectivement l'extraction d'information de surface et le résumé automatique de texte vu sous l'angle de l'extraction de phrases pertinentes d'un document. Le but de nos travaux a été d'explorer sur ces deux instances les apports potentiels des techniques d'apprentissage pour la recherche d'information textuelle.

Bien que nous ayons utilisé des approches totalement différentes pour traiter ces deux problèmes, les solutions que nous avons développées reposent toutes deux sur les capacités des systèmes d'apprentissage à exploiter automatiquement les ressources textuelles disponibles. Elles montrent que ces méthodes permettent d'aborder de nouvelles tâches, d'améliorer l'existant et de proposer des solutions génériques pouvant s'adapter à différentes tâches ou corpus.

Pour le premier problème nous avons développé des modèles stochastiques pour l'analyse de séquences, pour des problèmes de surlignage et d'extraction d'information de surface. Ces modèles sont basés sur des *perceptrons multi-couches* et des *modèles de Markov cachés*. Ils considèrent le texte comme une séquence de mots, à l'opposé des systèmes classique de RI, où un document est considéré comme un ensemble non ordonné d'index de mots. Ces modèles peuvent prendre en compte différentes contraintes qui traduisent soit des connaissances a priori sur la nature du corpus soit des besoins propres à la tâche. Ce type de système ouvre des perspectives pour l'automatisation de tâches d'extractions simples, et l'automatisation de certaines des tâches réalisées avec des systèmes d'extraction complets

Pour le résumé, nous avons développé deux techniques, l'une s'appuie sur l'interaction utilisateur et l'autre utilise de l'apprentissage non supervisé ou de l'apprentissage semi-supervisé. Cette dernière technique a des performances un peu plus faibles que celles obtenues par interaction, mais elle a l'avantage d'être totalement automatique. Elle a été implémentée et testée en utilisant deux modèles de discrimination (linéaire et logistique) et une adaptation de l'algorithme CEM. Nous avons donné la preuve de convergence de ces deux modèles. L'originalité de cette approche est que nos modèles adoptent une approche discriminante au lieu d'estimer des densités conditionnelles comme cela est le cas dans la plupart des méthodes basées sur des techniques non-supervisées et semi-supervisées.

Nous espérons que ces travaux contribueront à montrer aux communautés apprentissage et recherche d'information les nombreux intérêts qu'ils partagent et la complémentarité de compétence dont ils peuvent tirer profit.

Au-delà de cet intérêt général qui a motivé tout le cours de notre recherche, ces travaux ouvrent plusieurs directions :

Les modèles de séquence ont commencé à être appliqués à des tâches réelles d'extraction. Des systèmes basés sur des idées similaires à celles que nous avons développées ont été utilisés très récemment pour la segmentation et l'indexation de textes à usage de moteurs de recherche spécialisés. Ce sont des tâches relativement simples, mais qui illustrent bien l'apport de ce type de techniques. Ces méthodes peuvent être utilisées pour de nombreuses tâches nécessitant l'analyse automatique de séquences, toutefois, la crédibilité que l'on peut leur accorder reposera sur des tests qui nécessitent la constitution de bases de données qui n'existent pas actuellement, sur des développements complémentaires à ceux que nous avons effectués et surtout sur des analyses expérimentales poussées. L'interaction avec des systèmes d'extraction d'information dans le but d'automatiser leur développement est également un champ de recherche totalement ouvert qui prend une grande importance pour l'exploitation de certaines bases de données spécialisées, e.g. médicales. Ces développements nécessitent une coopération étroite des communautés extraction et apprentissage, qui n'a débuté qu'assez récemment.

L'apprentissage semi-supervisé apparaît très prometteur pour l'accès à l'information. Là aussi, beaucoup reste à faire pour exploiter le potentiel de ces techniques. Au niveau théorique il reste à construire une théorie de l'apprentissage semi-supervisé. Pour les algorithmes, nous pensons que la voie que nous avons explorée - à savoir l'utilisation de techniques discriminantes - permet de concevoir des méthodes économiques et robustes, mais là aussi très peu de travaux ont été réalisés. En ce qui concerne la démonstration de l'efficacité de ces techniques, il reste à la prouver expérimentalement en abordant un éventail de tâches différentes.

## Annexe

### Etude des décharges partielles

#### Représentation

La présence de décharges partielles se détecte en mesurant le déplacement de charges dans les conducteurs électriques, cette détection se fait en comptabilisant les quantités élémentaires relatives à une décharge sur la durée totale d'un essai. Une décharge partielle est caractérisée par son amplitude  $q_i$  et son emplacement  $\varphi_i$  sur une fenêtre de la phase. La représentation usuelle de ces décharges se fait via la distribution 3-d qui met en relation l'angle de la phase  $\varphi_i$ , la charge apparente  $q_i$  avec le nombre de décharges  $H_n(\varphi_i, q_j)$  (Figure 15 gauche). D'autres distributions peuvent être déduites de cette représentation (Figure 15 droite) : comme  $H_{qmax}(\varphi)$  la distribution maximum de charge,  $H_{qn}(\varphi)$  la distribution moyenne de charge,  $H_n(\varphi)$  la distribution du nombre de décharges.  $H(q)$  le nombre de décharges en fonction de la charge  $q$  et  $H(p)$  le nombre de décharges en fonction de l'énergie  $p$ .

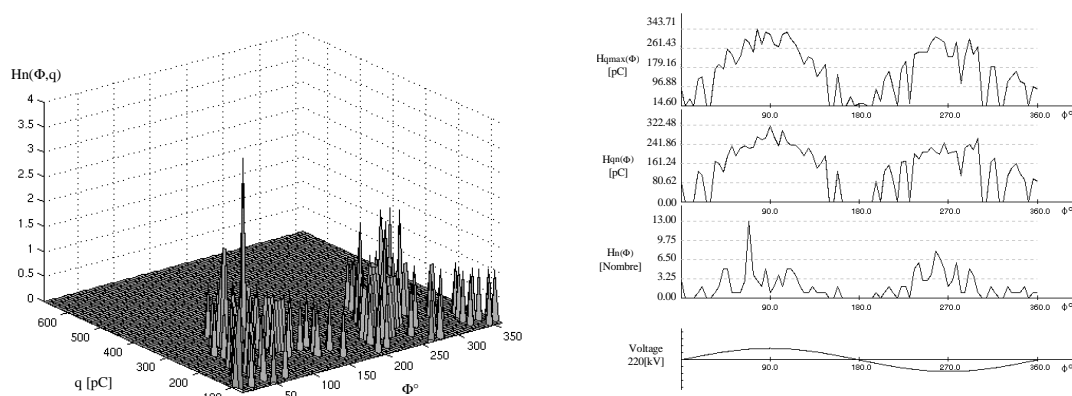


Figure 51. Gauche : la représentation 3-d d'une décharge, la côte  $n_{ij}$  représente le nombre de décharges ayant comme charge  $q_i$  et comme position  $\varphi_j$ . Droite : quelques distributions se déduisant de la représentation 3-d ; du haut en bas  $H_{qmax}(\varphi)$ ,  $H_{qn}(\varphi)$ ,  $H_n(\varphi)$ .

Les décharges partielles caractérisent des phénomènes aléatoires et elles sont influencées par plusieurs facteurs comme l'amplitude et la fréquence du voltage. Les mesures de ces décharges peuvent se détériorer par des interférences et le bruit intrinsèque des appareils. Ces formes 3-d sont alors complexes et leur caractérisation est un défi. Nous nous sommes intéressés à classer les décharges suivant 4 classes : les effets de couronnes (des pointes saillantes dans les électrodes), les effets de surfaces

(irrégularités de surface), les arcs électriques (des particules libres dans les isolants) et le bruit des instruments de mesures.

### *Pre-traitements*

Classiquement dans la littérature des systèmes de puissance, les caractéristiques issues de la représentation 3-d  $H_n(\varphi_i, q_j)$  sont utilisées pour décrire les décharges partielles. Nous avons combiné les caractéristiques les plus utilisées afin d'en donner une représentation plus riche. Ces caractéristiques sont : les statistiques simples sur les distributions  $H_{qmax}(\varphi)$ ,  $H_{qn}(\varphi)$ ,  $H_n(\varphi)$ ,  $H(q)$  et  $H(p)$ , la transformation de Fourier de  $H_{qn}(\varphi)$  et les mesures fractales de la forme 3-d  $H_n(\varphi_i, q_j)$ .

Nous allons d'abord décrire brièvement les méthodes d'au-dessus et ensuite nous allons présenter l'application des méthodes de Bagging et de Boosting sur ce problème en utilisant comme classifieur de base un perceptron multi-couches.

### *Opérateurs statistiques*

Des informations utiles sur les distributions  $H_{qmax}(\varphi)$ ,  $H_{qn}(\varphi)$ ,  $H_n(\varphi)$ ,  $H(q)$  et  $H(p)$  peuvent être inférées à partir de leurs moments. Les caractéristiques suivants ont été trouvées informatives pour la détection des décharges [GUL91], en indiquant  $\mu_i$  le moment d'ordre  $i$  de la variable  $x$  pour une distribution donnée : le *Skewness*,  $Sk = \mu_3/\mu_2^{3/2}$ , qui mesure l'asymétrie d'une distribution, le *Kurtosis*,  $Ku = \mu_4/\mu_2^{3/2} - 3$ , qui mesure l'aplatissement d'une distribution, le nombre de pics d'une distribution et l'asymétrie de la charge.

### *Transformation de Fourier*

La distribution normalisée de  $H_{qn}(\varphi)$  peut être estimée par la valeur moyenne de ces composantes spectrales dans le domaine des fréquences. L'estimation de la distribution moyenne de charges normalisées est donnée par

$$\tilde{H}_{qn}(\varphi) = \frac{a_0}{2} + \sum_{k=1}^{\infty} [\bar{a}_k \cdot \cos(k\varphi) + \bar{b}_k \cdot \sin(k\varphi)]$$

Où  $a$  et  $b$  sont les coefficients de Fourier dans le domaine des fréquence et  $\varphi$  indique l'angle de la phase. Nous avons utilisé les 8 premières composantes spectrales pour approximer la distribution de charge  $H_{qn}(\varphi)$ .

### *Caractéristiques Fractales*

Les mesures fractales permettent de calculer des caractéristiques globales d'une image. Nous avons utilisé deux mesures fractales sur les représentations 3-d des décharges ; la dimension fractale et la lacunarité [SAT95] qui quantifient respectivement le degré d'interruption et l'irrégularité d'une surface.



**Le maximum de vraisemblance d'une population  $X \sim \mathcal{N}_p(\mu, \Sigma)$**

Dans le cas où les données seraient issues d'une distribution normale  $X \sim \mathcal{N}_p(\mu, \Sigma)$  la fonction de densité des données est :

$$p(x/\mu, \Sigma) = (2\pi)^{-p/2} (\det \Sigma)^{-1/2} \exp[-\frac{1}{2} (x-\mu)^t \Sigma^{-1} (x-\mu)]$$

et le logarithme de la vraisemblance de mélange basé sur les  $m$  observations  $X = (x_1, \dots, x_m)$  s'écrit

$$\begin{aligned} L_M(\mu, \Sigma) &= \log P(X / \mu, \Sigma) = \log \left( \prod_{i=1}^m (2\pi)^{-p/2} (\det \Sigma)^{-1/2} \exp \left[ -\frac{1}{2} (x_i - \mu)^t \Sigma^{-1} (x_i - \mu) \right] \right) \\ &= \log \left( (2\pi)^{-mp/2} (\det \Sigma)^{-m/2} \exp \left[ -\frac{1}{2} \sum_{i=1}^m (x_i - \mu)^t \Sigma^{-1} (x_i - \mu) \right] \right) \\ &= -\frac{mp}{2} \log(2\pi) - \frac{m}{2} \log(\det \Sigma) - \frac{1}{2} \sum_{i=1}^m (x_i - \mu)^t \Sigma^{-1} (x_i - \mu) \end{aligned}$$

Posons  $\bar{x}$  et  $Var$  sont respectivement la moyenne et la variance estimée sur les données. Nous pouvons réécrire le troisième terme de  $L_M$  en y injectant  $\bar{x}$  et  $Var$ .

On a d'abord:

$$\begin{aligned} \sum_{i=1}^m (x_i - \mu)^t \Sigma^{-1} (x_i - \mu) &= tr \left[ \sum_{i=1}^m (x_i - \mu)^t \Sigma^{-1} (x_i - \mu) \right] = \sum_{i=1}^m tr \left[ (x_i - \mu)^t \Sigma^{-1} (x_i - \mu) \right] \\ &= \sum_{i=1}^m tr \left[ \Sigma^{-1} (x_i - \mu) (x_i - \mu)^t \right] = tr \left[ \sum_{i=1}^m \Sigma^{-1} (x_i - \mu) (x_i - \mu)^t \right] \\ &= tr \left[ \Sigma^{-1} \sum_{i=1}^m (x_i - \mu) (x_i - \mu)^t \right] \end{aligned}$$

Où  $tr$  est la fonction qui pour tout réel  $a$  donne,  $tr(a)=a$ . D'autre part en additionnant et soustrayant  $\bar{x}$ <sup>15</sup> dans le deuxième membre de l'égalité d'au-dessus il vient :

$$\begin{aligned} \sum_{i=1}^m (x_i - \mu)^t \Sigma^{-1} (x_i - \mu) &= tr \left[ \Sigma^{-1} \sum_{i=1}^m (x_i - \bar{x}) + (\bar{x} - \mu) (x_i - \bar{x}) + (\bar{x} - \mu)^t \right] \\ &= tr \left[ \Sigma^{-1} \sum_{i=1}^m (x_i - \bar{x}) (x_i - \bar{x})^t + m(\bar{x} - \mu) (\bar{x} - \mu)^t \right] \\ &= tr \left[ \Sigma^{-1} .m.[Var + (\bar{x} - \mu) (\bar{x} - \mu)^t] \right] \\ &= m. \left[ tr(\Sigma^{-1} .Var) + (\bar{x} - \mu)^t \Sigma^{-1} (\bar{x} - \mu) \right] \end{aligned}$$

Soit donc :

$$L_M(\mu, \Sigma) = -\frac{mp}{2} \log(2\pi) - \frac{m}{2} \log(\det \Sigma) - \frac{1}{2} \left[ tr(\Sigma^{-1} .Var) + (\bar{x} - \mu)^t \Sigma^{-1} (\bar{x} - \mu) \right]$$

<sup>15</sup> On a  $\bar{x} = \frac{1}{m} \sum_{i=1}^m x_i$ , et  $m.Var = \sum_{i=1}^m (x_i - \bar{x})(x_i - \bar{x})^t$

Maximiser  $L_M$  revient de minimiser d'une manière équivalente la fonction  $l^*$  :

$$l^*(\mu, \Sigma) = \log(\det \Sigma) + tr(\Sigma^{-1}.Var) + (\bar{x} - \mu)^t \Sigma^{-1} (\bar{x} - \mu)$$

$\Sigma$  est une matrice symétrique définie, positive il en est de même de son inverse  $\Sigma^{-1}$ , on a donc

$$(\bar{x} - \mu)^t \Sigma^{-1} (\bar{x} - \mu) \geq 0$$

Le troisième terme de  $l^*$  s'annule donc pour

$$\mu = \bar{x} = \frac{1}{m} \sum_{i=1}^m x_i$$

Pour cette valeur de  $\mu$  il reste à minimiser

$$l^*(\bar{x}, \Sigma) = \log(\det \Sigma) + tr(\Sigma^{-1}.Var)$$

Soit d'une manière équivalente minimiser la fonction

$$l^*(\bar{x}, \Sigma) - \log(\det Var) = tr(\Sigma^{-1}.Var) - \log \det(\Sigma^{-1}.Var)$$

Comme la matrice  $Var$  est symétrique définie positive on peut la décomposer comme un produit de ses racines :

$$Var = Var^{1/2}.Var^{1/2}$$

Soit,  $tr[\Sigma^{-1}.Var] = tr[\Sigma^{-1}.Var^{1/2}.Var^{1/2}] = tr[Var^{1/2}.\Sigma^{-1}.Var^{1/2}]$  et  $\det(\Sigma^{-1}.Var) = \det(Var^{1/2}.\Sigma^{-1}.Var^{1/2})$ , or la matrice  $A = Var^{1/2}.\Sigma^{-1}.Var^{1/2}$  est elle aussi symétrique, définie positive donc

$$tr[A] = \sum_{i=1}^p \lambda_i, \text{ et } \det(A) = \prod_{i=1}^p \lambda_i$$

où  $(\lambda_1, \dots, \lambda_p)$  sont les valeurs propres de  $A$  et de plus  $\forall i, \lambda_i \geq 0$ .  $l^*$  peut s'écrire alors :

$$\begin{aligned} l^*(\bar{x}, \Sigma) - \log(\det Var) &= \sum_{i=1}^p \lambda_i - \log\left(\prod_{i=1}^p \lambda_i\right) \\ &= \sum_{i=1}^p (\lambda_i - \log(\lambda_i)) \end{aligned}$$

Nous avons à choisir des  $\lambda_i$  de façon à minimiser cette équation, or la fonction  $h(x) = x - \log(x)$  n'a qu'un minimum global au point  $x = 1$ , et  $l^*$  est une somme de  $p$  fonctions  $h$ . Nous devons donc prendre  $\lambda_1 = \dots = \lambda_p = 1$ . Ce qui veut dire qu'à la valeur minimum de  $l^*$ ,  $A = Var^{1/2}.\Sigma^{-1}.Var^{1/2}$  est une matrice avec  $p$  valeurs propres tout égales à 1. Une telle matrice ne peut être que la matrice identité  $I_p$  d'où

$$\begin{aligned} Var^{1/2}.\Sigma^{-1}.Var^{1/2} &= I_p \text{ et donc } \Sigma^{-1} = Var, \text{ et} \\ \Sigma &= Var. \end{aligned}$$

Les valeurs  $\mu$  et  $\Sigma$  qui maximisent la vraisemblance  $L_M(\mu, \Sigma)$  s'obtiennent donc par l'estimation de la moyenne et de la variance sur les données.

### Le logarithme de la vraisemblance de classification

Supposons que les données (de dimension  $p$ ) sont obtenues aléatoirement par une densité de mélange de façon à ce que le nombre d'observations soit une distribution multinomiale. Posons  $T=(t_1, \dots, t_c)$  le vecteur indicateur de classe de l'exemple  $x$  tel que :

$$x \in C_k \Leftrightarrow t_k = 1 \text{ et } \forall h \neq k, t_h = 0.$$

On peut définir les poids de mélange  $\pi_k$  comme les probabilités d'appartenance aux classes  $\pi_k = p(t_k=1)$ , soit

$$p(T) = p(t_1, \dots, t_c) = \begin{cases} \pi_k, & \text{si } t_k = 1 \text{ et tous les autres } 0, \\ 0, & \text{sin on} \end{cases}$$

$$= \prod_{k=1}^c \pi_k^{t_k}$$

On peut écrire la probabilité jointe entre les données et les étiquettes comme :

$$p(x, T, \Theta) = p(T) \times p(x/T, \Theta)$$

$$= \pi_k \cdot p(x/P_k, \theta_k), \text{ si } t_k=1 \text{ et tous les autres } t_h=0$$

$$= \prod_{k=1}^c (\pi_k \cdot p(x/P_k, \theta_k))^{t_k}$$

La fonction de vraisemblance basée sur  $m$  observations  $x_1, \dots, x_m$  est alors:

$$V(P, \pi, \theta) = \prod_{i=1}^m p(x_i, T_i, \Theta)$$

$$= \prod_{i=1}^m \prod_{k=1}^c (\pi_k \cdot p(x_i/P_k, \theta_k))^{t_{ki}}$$

Le critère du maximum de vraisemblance pour la classification est définie comme le logarithme de  $V$ :

$$L_C(P, \pi, \theta) = \sum_{k=1}^c \sum_{i=1}^m t_{ki} \cdot \log(\pi_k \cdot p(x_i/P_k, \theta_k))$$

## Système d'Aide au Résumé Automatique (mode interactif)

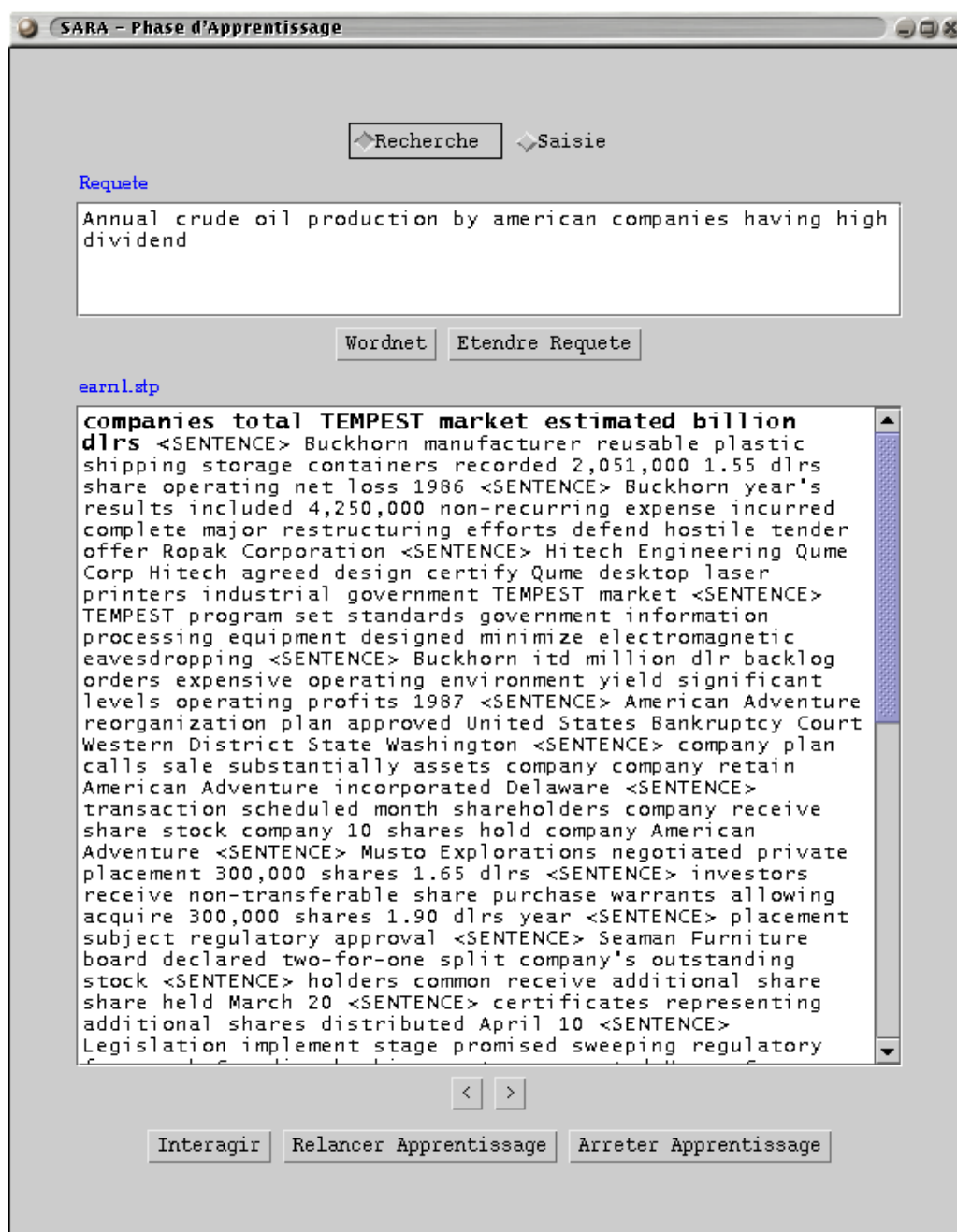


Figure 52. Projection de la requête sur les phrases des documents

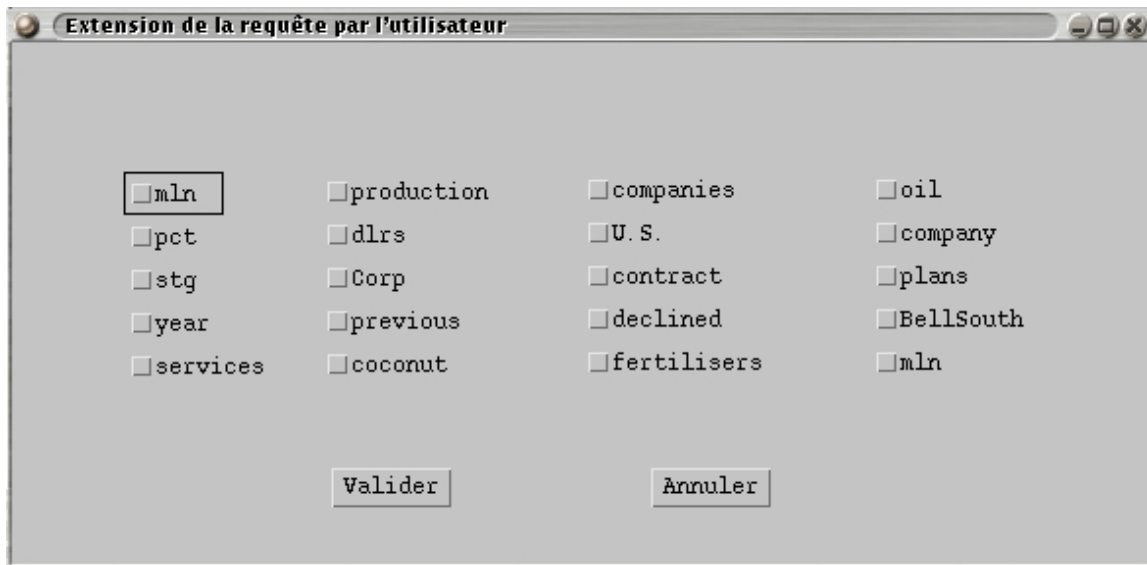


Figure 53. Extension de la requête par l'utilisateur

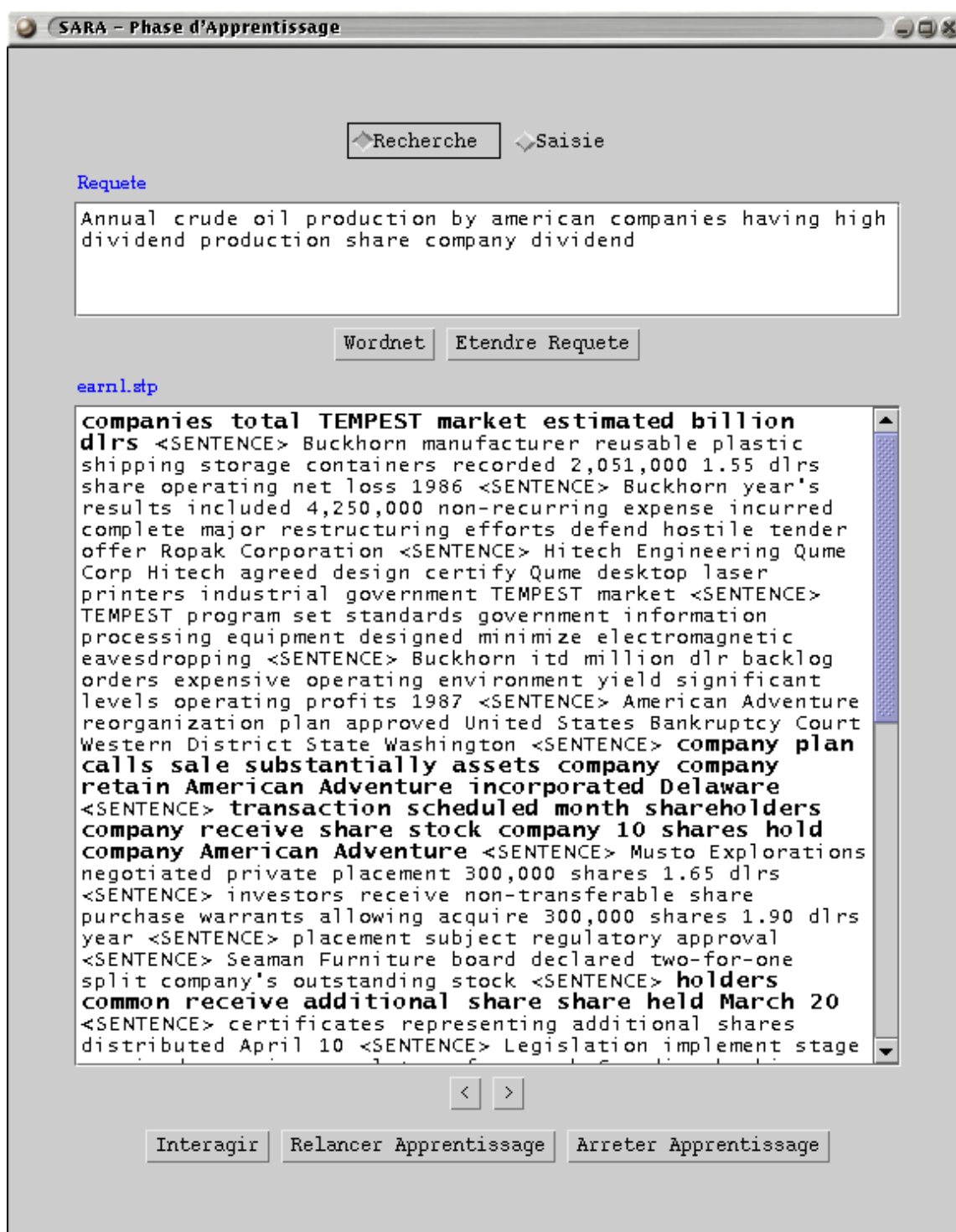


Figure 54. Nouvelle projection de la requête étendue sur les phrases du document

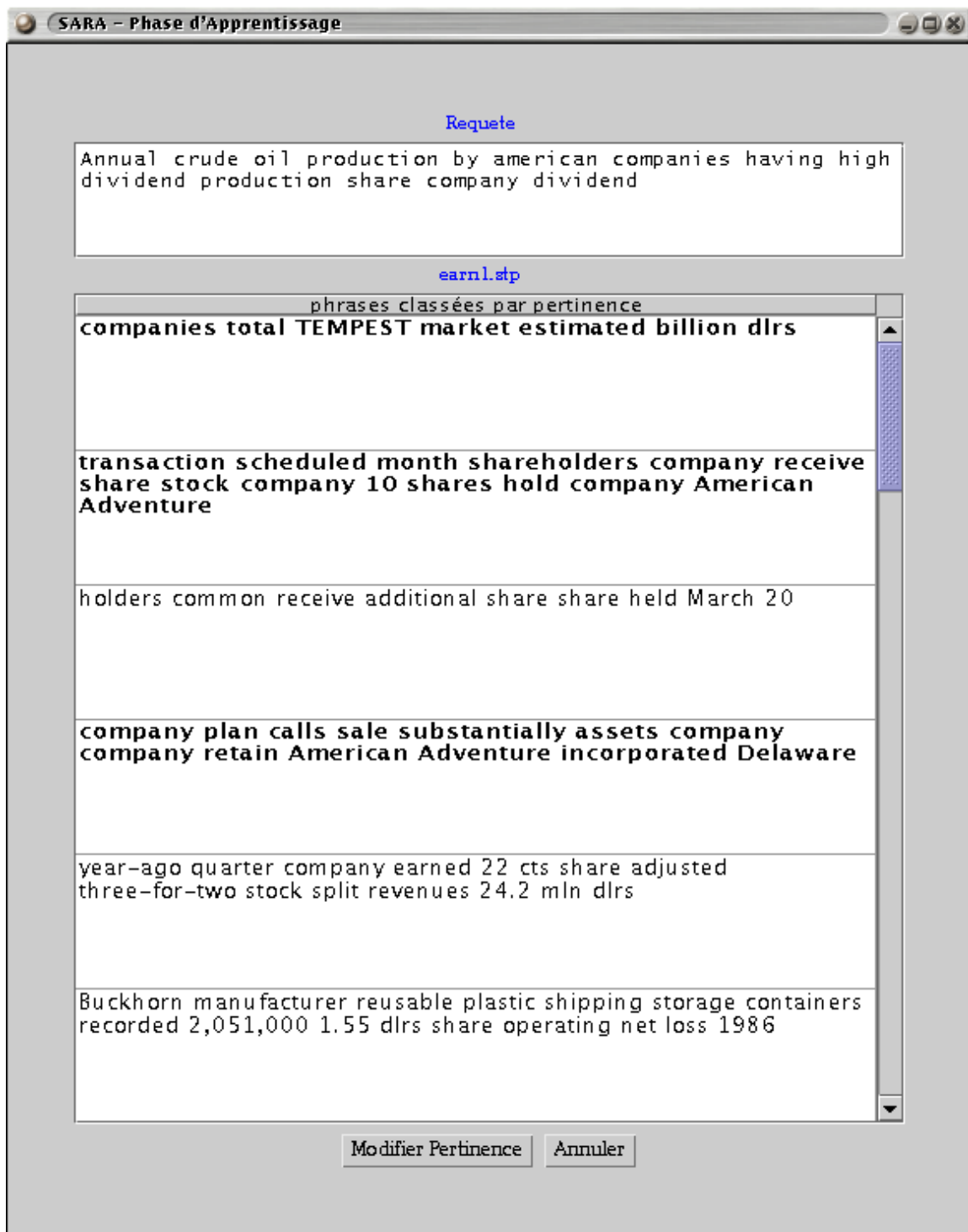


Figure 55. Interaction utilisateur : modification de pertinence des phrases

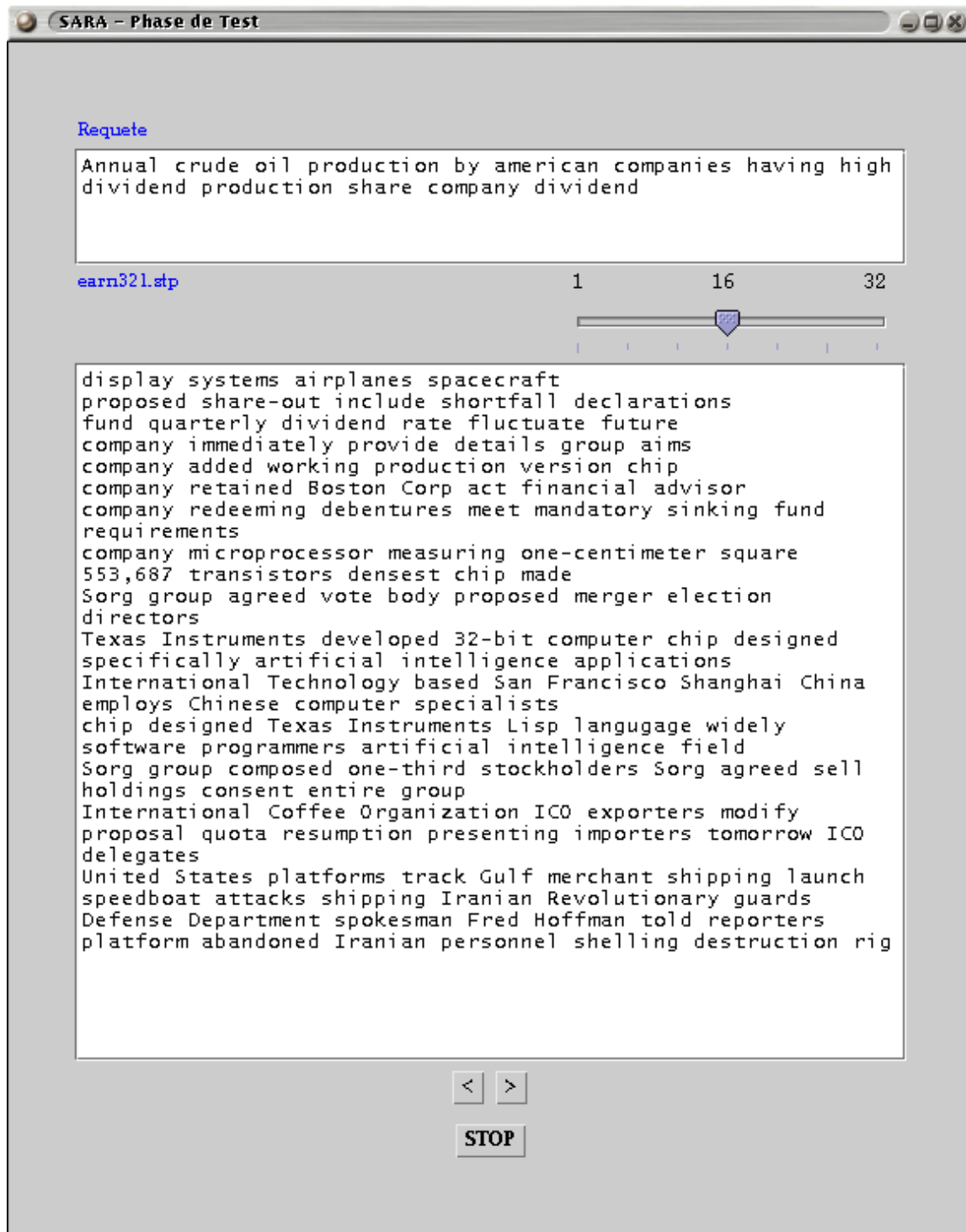


Figure 56. Réponse du système après apprentissage et interaction



## **Bibliographie Personnelle**

### **Chapitre de livre avec comité de lecture**

[2000] P. Gallinari, H. Zaragoza, M.-R. Amini, « *Apprentissage et Données Textuelles* ». à paraître.

### **Conférences internationales avec comité de lecture**

[2001] M.-R. Amini, P. Gallinari, « *Automatic Text Summarization using Unsupervised and Semi-supervised Learning* », 12<sup>th</sup> European Conference on Machine Learning, ECML'01, à paraître.

[2001] M.-R. Amini, P. Gallinari, « *Learning for Text Summarization using Labeled and Unlabeled Sentences* », 11<sup>th</sup> International Conference of Artificial Neural Networks, ICANN'01, à paraître.

[2001] M.-R. Amini, P. Gallinari, « *Self-Supervised Learning for Automatic Text Summarization by Text-span Extraction* », 23<sup>rd</sup> BCS European Annual Colloquium on Information Retrieval. ECIR'01, pp. 55--63, Darmstadt, Allemagne.

[2000] M.-R. Amini, H. Zaragoza, P. Gallinari, « *Learning for Sequence Extraction Tasks* », 6th Conference on "Content-Based Multimedia Information Access" – Recherche d'Informations Assistée par Ordinateur. RIAO'2000, édition Cid, pp. 476--490.

[1999] M.-R. Amini, H. Zaragoza, P. Gallinari, « *Stochastic Models for Surface Information Extraction in Texts.* », 9th International Conference of Artificial Neural Networks. ICANN'99, édition IEE, pp. 892--897.

[1999] M.-R. Amini, H. Zaragoza, P. Gallinari, « *Sequence Models for Automatic Highlighting and Surface Information Extraction.* », 21st Conference of the Information Retrieval Specialist Group. IRSG'99, pp. 85--91.

[1998] M.-R. Amini, P. Gallinari, F. d'Alché-Buc, F. Bonnard, E. Fernandez., « *Automated Statistical Recognition of Partial Discharges in Insulation Systems.* », 8th International Conference of Artificial Neural Networks. ICANN'98, édition Springer, pp. 701--706.

### **Workshop international avec comité de lecture**

[2000] M.-R. Amini, « *Interactive Learning for Text Summarization* », PKDD'2000/MLTIA'2000 Workshop on Machine Learning and Textual Information Access, 2000, pp. 44--52, Lyon France.

### **Conférence nationale avec comité de lecture**

[2000] M.-R. Amini, H. Zaragoza, P. Gallinari, « *Utilisation des Réseaux de Neurones pour l'Analyse de Séquences dans les Textes* », Conférence d'Apprentissage, CAp'2000, édition Hermès, pp. 21--30.



## Liste des Figures

### Liste des Figures

- Figure 1. La fonction ET (à gauche) est linéairement séparable mais pas la fonction XOR (à droite).13
- Figure 2. Modèle linéaire des réseaux de neurones. .... 13
- Figure 3. Différentes fonctions de transfert (a) Fonction de Heaviside, (b) Fonction linéaire par morceaux, (c) Fonction sigmoïde. .... 14
- Figure 4. Architecture d'un perceptron composé de quatre composantes principales : la rétine, les fonctions d'association, les poids synaptiques et l'unité à seuil. .... 16
- Figure 5. L'illustration de la convergence du perceptron pour un problème simple de classification à deux classes et quatre exemples. a) pour un vecteur de poids, l'hyperplan séparateur est représenté par celui ayant la direction perpendiculaire à ce vecteur, b) la correction de poids se fait en soustrayant le vecteur de poids par le vecteur représentant le premier exemple mal classé, c) en répétant cette procédure le perceptron trouve le bon hyperplan séparateur. .... 17
- Figure 6. Problème de classification linéaire à deux classes et solutions trouvées par le perceptron (en pointillée) et par l'Adaline (en gras). .... 18
- Figure 7. Architecture d'un perceptron multi-couches à une couche cachée. Les paramètres de biais sont introduits par des poids liés à une unité d'entrée supplémentaire ayant la valeur fixée  $x_0=1$ . .... 19
- Figure 8. Les trois types de MMC les plus répandues. a) un MMC ergodique à 4 états. b) un MMC gauche-droite à 4 états. c) un MMC parallèle gauche-droite à 6 états. .... 22
- Figure 9. Illustration du mécanisme de calcul de la probabilité de la séquence d'observation par l'algorithme *forward*. Le calcul des  $\alpha_{t+1}(j)$  nécessite uniquement de connaître les  $\alpha_t(i)$  et les  $a_{ij}$ ,  $i=1, \dots, l$ , où les connexions  $a_{ij}$  représentent les probabilités de transition entre états.23
- Figure 10. Illustration du mécanisme de calcul de la probabilité de la séquence d'observation par l'algorithme *backward*. .... 25
- Figure 11. Les hyperplans pour un problème de classification linéairement séparable. Les points à support vecteurs sont encerclés. .... 28
- Figure 12. Les hyperplans linéaires pour un problème de classification non-linéairement séparable. Lorsqu'il y a une erreur sur un exemple, cet exemple est considéré comme un vecteur support dont sa distance avec l'hyperplan de sa vraie classe est  $-\xi/\|w\|$ . .... 29
- Figure 13. En plongeant les données dans un espace de plus grande dimension, on peut simplifier la tâche de classification. .... 31
- Figure 14. Une illustration schématique de la notion de biais et de variance. Les cercles représentent un ensemble de points générés par une fonction  $h(x)$  (en pointillée) avec du bruit additionnel. Le but est d'approximer  $h(x)$  le plus près que possible. Si l'on modélise les données par une fonction fixe  $f(x)$ , le biais est très élevé tandis que la variance est nulle. 44
- Figure 15. Comme dans la figure 14, mais le modèle utilisé est un interpolant exact des données. Dans ce cas le biais est faible mais la variance est élevée. .... 44
- Figure 16. Courbes d'erreur (en apprentissage et en test) et distribution des marges pour le Boosting et le Bagging utilisant un PMC sur la base de décharges partielles. Les courbes du haut montrent les erreurs en apprentissage (trait plein) et en test (trait pointillé) en fonction du nombre de classificateurs combinés. Les courbes du bas montrent les distributions des marges

	cumulées de la base d'apprentissage, pour chacun de ces algorithmes après 2 (en trait pointillé) et 100 itérations (en trait plein).....	51
Figure 17.	Une représentation graphique montrant le fonctionnement de l'algorithme EM. L'axe des abscisses représente les valeurs possibles de l'ensemble des paramètres $\Theta$ et l'axe des ordonnées donne le logarithme de la vraisemblance des données. L'algorithme EM calcule la fonction $Q(\Theta, \Theta^{(j)})$ en utilisant l'estimation courante $\Theta^{(j)}$ et donne la nouvelle $\Theta^{(j+1)}$ comme le point maximum de $Q(\Theta, \Theta^{(j)})$ .....	55
Figure 18.	Mécanisme de l'algorithme de décision dirigée dans le cas adaptatif. Lorsqu'on est en présence d'un exemple non-étiqueté $x$ , le système va lui attribuer une étiquette $t$ , en seuillant la sortie calculée pour cet exemple. Dans la deuxième étape il va apprendre la probabilité a posteriori des classes $p(t/x)$ .....	68
Figure 19.	apprentissage supervisé vs apprentissage auto-supervisé.....	70
Figure 20.	Le réseau pour l'apprentissage des étiquettes des vecteurs de références. Les vecteurs poids des neurones de la couche cachée représentent les vecteurs de références et les vecteurs poids connectant les neurones de la couche cachée et les neurones de sorties représentent les classes. Dans ce schéma il y a 3 classes et deux modalités. ....	71
Figure 21.	Si une modalité reçoit un exemple de sa distribution de classe A, la modalité 2 reçoit un exemple de sa propre distribution de classe A. Sans aucune information sur la classe des exemples, les deux classifieurs doivent déterminer le placement des frontières $b_1$ et $b_2$ appropriées. ....	72
Figure 22.	Un rapport sur un attentat terroriste et une fiche d'information extraite.....	87
Figure 23.	Structure d'un système d'Extraction de l'Information. L'exemple consiste à extraire à partir du document initial les informations concernant les mouvements de personnels sous la forme d'un formulaire qui est représenté en bas de la figure. ....	88
Figure 24.	La structure d'un réseau Naïve Bayes.....	94
Figure 25.	principales composantes d'un système de RI, A) Indexation, B) Appariement, C) Décision D) retour soit par interaction utilisateur soit de façon automatique .....	96
Figure 26.	MMC ergodique à deux états pour la recherche documentaire .....	101
Figure 27.	La courbe de rappel et de précision tracée à partir de l'exemple du tableau 1.....	105
Figure 28.	Graphe de segmentation et des liens thématiques d'un document. Chaque paragraphe est représenté par un numéro, ceux-ci sont positionnés dans l'ordre d'apparition dans le texte. Seuls sont reliés les paragraphes dont la similarité est supérieure à un seuil [SAL96]. ...	115
Figure 29.	Méthode de <i>TextTiling</i> , le graphe indique un score entre segments adjacents du texte et les lignes verticales indiquent les frontières entre ces segments.....	117
Figure 30.	Modèle de Markov caché pour la segmentation de texte. ....	117
Figure 31.	Schéma général d'un système de résumé automatique suivant l'approche « mesures de similarité ».....	122
Figure 32.	Les segments de phrases et leur relation rhétorique.....	123
Figure 33.	La structure rhétorique du discours.....	124
Figure 34.	Un exemple de résumé manuel (gauche) et un document (droite). L'alignement des passages est montré sur la figure. Aucun passage du document ne correspondent à la dernière ligne du résumé. ....	127
Figure 35.	Comportement de l'algorithme d'extraction de résumé de Marcu.....	128
Figure 36.	Un paragraphe (haut gauche) est comparé avec un patron (gauche bas) pour obtenir les étiquettes des mots (haut droite). Dans le patron nous avons indiqué en gras les deux champs que nous utilisons : POST et PER_NAME. Pour la tâche d'extraction les mots d'étiquettes PERSONNE sont montrés en gras et les mots d'étiquettes POSITION sont montrés en italiques. Pour la tâche de surlignage les termes d'étiquettes PERSONNE et POSITION sont montrés en gras. Les mots non-pertinents sont montrés en normal. ....	146
Figure 37.	Un modèle MMC correspondant à la topologie $\langle I/Per^{(3)}/Pos^{(3)} \rangle$ pour la tâche d'extraction (le modèle de durée n'est pas montré). Les distributions de probabilité associées aux états correspondent aux différentes classes, Personne (Per), Position (Pos) et non-pertinent (I).147	147

Figure 38. Courbes de Précision-Rappel obtenues par le PMC sans utilisation de contraintes sur les sorties. Les termes sont représentés par leur mesure $U$ (gauche) et leur mesure $U$ plus utilisation d'informations syntaxiques (droite).....	149
Figure 39. Les courbes de précision-rappel obtenues avec les modèles $U_{PMC}$ (ligne du haut) and $W_{MMC}$ (ligne du bas) pour la tâche de surlignage.....	150
Figure 40. Les courbes de Précision-Rappel obtenues avec les modèles $U_{PMC}$ et $W_{MMC}$ pour l'extraction des classes Position (gauche) et Personne (droite).....	151
Figure 41. Intervalle de confiance t-bootstrap à 95%, calculer pour la tâche de l'Extraction sans (a) et avec (b) utilisation de grammaire $\langle IR^{(3)} \rangle$ avec le modèle $U_{PMC}$ .....	153
Figure 42. Projection de la requête sur les phrases des documents, les phrases dont la similarité avec la requête est supérieure à un seuil calculé, sont indiquées en gras, elles seront proposées à l'utilisateur comme pertinentes. ....	162
Figure 43. Les phrases jugées pertinentes à une étape $j$ sont présentées à l'utilisateur dans l'ordre de leur pertinence et celui-ci interagit avec le système en indiquant pour certaines d'entre elles quelles sont celles qu'il juge pertinentes ou non-pertinentes. ....	163
Figure 44. Décision bayésienne pour un problème de classification à deux classes normales de même variance covariance. ....	168
Figure 45. Modèle d'apprentissage basé sur l'algorithme CEM-logistique. ....	175
Figure 46. Gauche. Distribution de la longueur des résumés en nombre de phrases, droite. Distribution de la longueur des résumés en nombre de mots. ....	176
Figure 47. La simulation de l'interaction se fait en deux étapes. A l'étape ① le classifieur apprend en mode non-supervisé grâce à un premier étiquetage obtenu par le système de base. A l'étape ②, on change les étiquettes des phrases qui sont dans le sous-ensemble d'interaction.....	179
Figure 48. La précision du système en fonction du degré d'interaction. L'axe des abscisses représente le pourcentage de documents dans la base d'apprentissage utilisés pour l'interaction. ....	179
Figure 49. Les courbes de rappel et de précision pour le système de base (carré), apprentissage non-supervisé (étoile), apprentissage semi-supervisé (triangle), apprentissage interactif (croix) et apprentissage supervisé (cercle). Le classifieur utilisé est le classifieur avec un neurone sigmoïde. ....	182
Figure 50. Comparaison des courbes de rappel et de précision entre l'algorithme de CEM-régression et la CEM-logistique dans le cas de l'apprentissage non-supervisé (a) et de l'apprentissage semi-supervisé (b). ....	183
Figure 51. Gauche : la représentation 3-d d'une décharge, la côte $n_{ij}$ représente le nombre de décharges ayant comme charge $q_i$ et comme position $\phi_j$ . Droite : quelques distributions se déduisant de la représentation 3-d ; du haut en bas $H_{qmax}(\phi)$ , $H_{qn}(\phi)$ , $H_n(\phi)$ .....	191
Figure 52. Projection de la requête sur les phrases des documents .....	196
Figure 53. Extension de la requête par l'utilisateur.....	197
Figure 54. Nouvelle projection de la requête étendue sur les phrases du document .....	198
Figure 55. Interaction utilisateur: modification de pertinence des phrases.....	199
Figure 56. Réponse du système après apprentissage et interaction .....	200



## Liste des Tables

Tableau 1. Un exemple jouet pour le calcul de la précision et du Rappel pour une requête $q$ donnée et pour un nombre croissant de réponses. ....	104
Tableau 2. La sélection de variables pour le surlignage. ....	142
Tableau 3. Les performances de $U_{MMC}$ et de $U_{PMC}$ pour le surlignage et l'extraction. Dans le premier cas, la moyenne est par rapport aux classes pertinent et non-pertinent. Tandis que dans le second elle est par rapport aux classes Position, Personne et non-pertinent. Dans chaque colonne les meilleures performances sont en gras. ....	148
Tableau 4. Les Performances des modèles $U_{PMC}$ , $U_{MMC}$ et $W_{MMC}$ pour le surlignage et l'extraction de surface pour quatre grammaires. Dans le cas de surlignage, la moyenne est sur les classes pertinent et non-pertinent tandis que dans le cas de l'extraction elle est sur les classes Position, Personne and non-pertinent. Les meilleures performances sont en gras. ....	149
Tableau 5. Performance Moyenne et Ecart type du modèle $U_{PMC}$ obtenue par Bootstrap dans le cas de surlignage avec $B = 40$ . ....	152
Tableau 6. Caractéristiques de la base de données Reuters et des résumés associés. ....	175
Tableau 7. Comparaison entre le système de base et les différentes techniques d'apprentissage avec l'algorithme CEM-régression utilisant un neurone linéaire comme classifieur. Les performances sont sur la base test. ....	180
Tableau 8. Comparaison entre deux modèles : linéaires et Machine à Vecteurs Supports dans le cas de l'apprentissage semi-supervisé avec l'algorithme <i>CEM-régression</i> . Les performances sont sur la base de test. ....	181
Tableau 9. Comparaison entre les fonctions de discrimination (7.7) et (7.9) dans les deux cas d'apprentissage semi-supervisé et non-supervisé. ....	181





## Liste des Algorithmes

Algorithme 1. Algorithme de rétro-propagation du gradient dans le cas de l'apprentissage en-ligne.	21
Algorithme 2. Algorithme <i>forward</i> pour le calcul de $p(x/\lambda)$ .	23
Algorithme 3. Algorithme de <i>Viterbi</i> pour trouver la séquence d'états optimale.	24
Algorithme 4. Algorithme <i>backward</i> pour le calcul des $\beta$ .	25
Algorithme 5. L'algorithme de Bagging.	46
Algorithme 6. L'algorithme d'AdaBoost (M1)	47
Algorithme 7. L'algorithme d'AdaBoost (M2)	48
Algorithme 8. L'algorithme de EM	55
Algorithme 9. Algorithme EM pour l'estimation des paramètres d'un mélange de densités gaussiennes.	57
Algorithme 10. L'algorithme CEM	58
Algorithme 11. Algorithme EM pour l'estimation des paramètres d'un mélange de densités dans le cas de supervision imparfaite.	60
Algorithme 12. Algorithme <i>décision dirigée</i> .	69
Algorithme 13. L'algorithme de Co-Boosting.	74
Algorithme 14. L'algorithme de Co-Training	76
Algorithme 15. L'algorithme semi-supervisé basé sur l'algorithme EM	79
Algorithme 16. L'Algorithme de construction de Marcu pour construire des résumés extraits à partir de résumés manuels.	127
Algorithme 17. Algorithme de Bootstrap	152
Algorithme 18. Algorithme informel pour l'apprentissage non-supervisé.	165
Algorithme 19. Algorithme CEM dans le cas semi-supervisé.	167
Algorithme 20. L'algorithme CEM-régression en non supervisé.	171
Algorithme 21. L'algorithme CEM-logistique - cas non-supervisé	173