Classification du texte numérique et numérisé. Approche fondée sur les algorithmes d'apprentissage automatique

Karim Sayadi

▶ **To cite this version:**

Karim Sayadi. Classification du texte numérique et numérisé. Approche fondée sur les algorithmes d'apprentissage automatique. Algorithme et structure de données [cs.DS]. Université Pierre et Marie Curie - Paris VI, 2017. Français. NNT : 2017PA066079 . tel-01613499

HAL Id: tel-01613499
https://theses.hal.science/tel-01613499

Submitted on 9 Oct 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université Pierre et Marie Curie

ÉCOLE DOCTORALE 564 PIF

Physique en Ile-de-France

T H È S E

pour obtenir le titre de

Docteur en sciences

de l'Université de Pierre et Marie Curie

Mention : Informatique

Présentée et soutenue par

Karim Sayadi

Classification du texte numérique et numérisé. Approche fondée sur les algorithmes d'apprentissage automatique.

Thèse dirigée par le Professeur Marc Bui

préparée au laboratoire CHArt EA4004 EPHE, Paris

Date de soutenance 28/03/2017

Jury :

Rapporteurs :	DR. CNRS. Thierry Poibeau	-	CNRS-ENS, Paris, France.
	DR. IE ILSP. Vassilis Katsouros	-	Institut ILSP, Athènes, Grèce.
Directeur :	Pr. Marc Bui	-	Université Paris 8-EPHE, France.
Président :	Pr. Jean Daniel Zucker	-	Université de Paris 6, France
Examinateurs :	Pr. Marcus Liwicki	-	Université de Fribourg, Suisse
	Pr. Vu Duong	-	Institut JVN, Ho Chi Minh ville, Vietnam
	MCf (HDR). Sofian Ben Amor	-	Université de Versailles, France

Le succès est le nectar de l'échec.

Remerciements

La thèse est un semblant d'une aventure solitaire, est un fruit d'un environnement riche en interactions. J'ai eu la chance d'être en même temps un étudiant de l'université de Pierre et Marie Curie et un doctorant du laboratoire CHArt à l'École Pratique des Hautes Études. Je souhaiterais remercier mon directeur de thèse Marc Bui. Marc m'a appris à faire de la recherche, à écrire mes premiers papiers, à présenter mes résultats, à préparer mes premiers cours, et à rédiger. Il m'a donné beaucoup de liberté pour poursuivre mes idées et il m'a supporté pour les présenter dans de nombreuses conférences. Je suis reconnaissant d'avoir été son étudiant et cela depuis le Master.

Je souhaiterais remercier mon comité de thèse, le président Jean Daniel Zucker, les rapporteurs Thierry Poibreau, Vassilis Katsouros, les examinateurs Marcus Liwicki, Vu Duong et Sofian Ben Amor.

Je souhaiterais remercier également les responsables du programme doctoral international IRD-UPMC, Christophe Cambier, Nicolas Marilleau, Edith Perrier, Serge Stinckwich, Jean Daniel Zucker, pour m'avoir fait confiance et rendu possible le déroulement de cette thèse.

Je souhaiterais remercier l'équipe du laboratoire d'information et technologie de l'université polytechnique de Hanoi, au Vietnam : Ha Quoac Trung, Linh Truong Dieu, Ngo Hong Son. Je souhaiterais également remercier l'équipe de l'institut Jhon Von Neumann de l'université de Ho Chi Minh ville, au Vietnam, dirigée par Vu Duong, merci à Lan Phan, Nguyen Hoang, Nguyen Duc, An Mai. Merci de m'avoir acceuilli et s'être assuré que mes premiers longs séjours en Asie se passent dans les meilleures conditions.

Je souhaiterais remercier l'équipe du laboratoire de l'analyse des documents, images et voix (DIVA) à l'université de Fribourg en Suisse. Merci à Rolf Ingold de m'avoir accueilli. Merci à Marcus Liwicki de m'avoir accueilli. Il m'a soutenu et dirigé dans mes premiers travaux sur l'analyse des documents anciens. Je ne peux exprimer que ma reconnaissance pour mes collègues à Fribourg : Mathias Seuret, Fotini Simistira, Marcel Würsch, Angelika Graz. Je souhaiterais également remercier Andreas Fischer pour ses conseils sur les travaux en commun avec son ancien étudiant Mansour Hamidi. Je souhaiterais remercié Daniel Stoekl de L'EPHE pour m'avoir introduit aux humanités computationnelles et m'avoir fait découvrir un monde riche en collaborations.

Je souhaiterais remercier les directeurs du laboratoire CHArt, Charles Tijus et François Jouen, les membres Joëlle Provasi, Isabelle Carchon et coté administration Céline Butelle. Je souhaiterais remercier mes collègues au laboratoire CHArt, les anciens qui m'ont guidé dans mes premiers pas : Pierre Leenhardt, Murat Ahat, Ke

Liang, Coralie Petermann, Vigile Hoareau, Omar Bennani et Sofian Ben Amor qui m'a introduit à ce laboratoire. Les nouveaux : Julio Labode, Anna Scius-Bertrand, Jérémie Bosom et enfin Quang Vu Bui avec qui j'ai rédigé mes premiers papiers et avec qui j'ai passé de longues heures devant l'écran à coder.

Enfin et surtout, je souhaiterais exprimé ma plus grande reconnaissance et gratitude à ma famille, oncles et tantes, cousins et cousines, spécialement à mes parents Salamet et Fethi et grands-parents, j'ai de la chance de vous avoir. Merci à mes soeurs Myriam et Rahma. Merci à Rim ma compagne et amie de toujours pour tout le support.

Table des matières

1 Introduction **9**

2 Contexte **15**
2.1 Notion d'algorithme d'apprentissage : le cadre théorique 15
2.2 Apprentissage contextualisé pour traiter des textes 17
2.2.1 Apprentissage supervisé 17
2.2.2 Apprentissage non supervisé 18
2.2.3 Sur-apprentissage et sous-apprentissage 18
2.3 Mise en oeuvre de l'apprentissage 19
2.4 Analyse de textes dans le cadre des humanités computationnelles . . 20
2.4.1 Discrimination et classification des documents 20
2.4.2 Analyse des images de textes 22

I Analyse automatique des textes numériques pour la classification thématique **25**

3 Structuration thématique des corpus de textes **27**
3.1 Le problème de la classification des textes 28
3.1.1 Mathématisation des corpus de textes et modéles de représentation thématique 29
3.2 L'allocation latente de Dirichlet 32
3.2.1 Caractérisation de notre approche par les modèles graphiques probabilistes 32
3.2.2 Présentation de notre approche 33
3.2.3 Opérationnalisation de notre approche 36
3.3 La prétopologie comme modèle de proximité 40
3.3.1 Les espaces prétopologiques comme espace sémantique 40
3.3.2 Les fermés comme outils de caractérisation de la structure . . 42
3.3.3 Mesures d'évaluation des modèles non supervisés de classification de textes 43

4 Classification des textes et gestion du passage à l'échelle **47**
4.1 Classification multicritères 48
4.1.1 Structuration des documents avec LDA 49
4.1.2 Association des documents par des relations binaires 49
4.1.3 Construction de la fonction d'adhérence 50
4.1.4 Construction d'une distance avec l'application de l'adhérence 52
4.2 Gestion du passage à l'échelle 64
4.2.1 Modèle synchrone pour la parallélisation de LDA 67

4.2.2 Implémentation parallèle de LDA avec Spark 69
4.2.3 Application aux données et évaluation 72

5 Classification hiérarchique des pages web avec une analyse thématique catégorielle 77
5.1 Version semi-supervisée de LDA 79
5.2 Utilisation des forêts aléatoires 80
5.3 Application et analyse 82

6 Construction de corpus annotés pour le dialecte tunisien 87
6.1 Jeu de données proposées et statistiques 88
6.2 Sélection de caractéristiques et algorithmes de classification 90
6.3 Résultats 92
6.4 Discussion de l'effet du nombre de caractères 93

II Analyse automatique des textes manuscrits 97

7 Identification des scripts dans les manuscrits anciens 99
7.1 Introduction 99
7.1.1 Position du problème 99
7.1.2 Spécification de notre jeu de données 102
7.1.3 Notre approche pour la classification des scripts 103
7.2 Méthode de travail 106
7.2.1 Rappels sur les réseaux de neurones artificiels 107
7.2.2 Les réseaux de neurones à convolutions (CNN) 110
7.2.3 Les auto-encodeurs (AE) 111
7.2.4 Les auto-encodeurs convolutif empilés (SCAE) 112
7.2.5 Architectures 114
7.3 Construction d'un référentiel empirique obtenu 115
7.3.1 Histogramme de gradients orientés 115
7.3.2 Classification 117
7.3.3 Résultats 118
7.4 Contribution pour l'apprentissage non supervisé de scripts avec les auto-encodeurs 119
7.4.1 Apprentissage des caractéristiques 120
7.4.2 Classification 124
7.4.3 Analyse des résultats 128
7.5 Contribution pour l'apprentissage supervisé de scripts avec les architectures à convolution 130
7.5.1 Sur l'importance de l'initialisation des paramètres 132
7.5.2 Classification 135
7.5.3 Analyse des résultats 138

8 Conclusion et perspectives **141**
8.1 Conclusion 141
8.2 Perspectives 142

A Les défis de l'apprentissage automatique **145**
A.1 Apprentissage supervisé 145
A.2 Formalisme de Vapnik et Chervonenkis 147
A.2.1 Cadre général 148
A.2.2 La dimension de Vapnik et Chervonenkis (VC) 149
A.2.3 Séparateurs à Vaste Marge (SVM) 152
A.3 Formalisme PAC 152
A.3.1 Cadre général 153
A.3.2 Algorithme PAC-apprenable 154
A.3.3 Le compromis Biais-Variance 155

B Réseaux bayésiens pour la modélisation des textes **157**
B.1 Les réseaux bayésiens 157
B.1.1 Raisonnement bayésien 158
B.2 Code de l'implémentation de l'allocation latente de Dirichlet 165

Bibliographie **171**

Liste des tableaux

3.1 La table de contingence. $n_{ij} = |P_i \cap Q_j|$ est le nombre de d'objets en commun entre le groupe de données P_i et Q_i. 44

4.1 Distribution des mots dans les thèmes ϕ et les utilisateurs qui relatent les thèmes en question 56
4.2 Distribution des thèmes par utilisateur 56
4.3 Classification des documents basée sur leurs thèmes majeurs 57
4.4 Sortie de l'algorithme de *K-means* en utilisant la distance de Hellinger 58
4.5 Result from *K-means* algorithm using Hellinger distance and MCPTM 58
4.6 Les résultats de *K-means* utilisant la distance de Hellinger sur le groupe numéro 13 (89 utilisateurs). 60
4.7 Résultats de l'analyse de similarité entre les sorties des K-moyennes initialisés avec différentes mesures de distances. E, est la distance d'Euclide, H la distance de Hellinger, P est la distance de l'adhérence. 62
4.8 Description des quatre jeux de données utilisés pour l'évaluation du passage à l'échelle. 72

5.1 Investigation des paramètres des forêts aléatoires sur le taux de classification hiérarchique selon plusieurs stratégies. 84

6.1 Jeux de données existants ordonnés par leur taille. 88
6.2 Exemples des mots clés/*hashtags* utilisés lors de la collecte des données sur Twitter. 88
6.3 Exemple des tweets annotés. 89
6.4 Statistiques du contenu du jeu de données annoté. 90
6.5 Sélection des caractéristiques sur différents n-grammes avec le ratio de sélection. 90
6.6 Les résultats des classifieurs entraînés avec des caractéristiques du dialecte Tunisien. L'abréviation g est pour -gramme (le modèle de langage) et l'abréviation c pour classes. 93

7.1 Description du jeu de données utilisé. Les scripts qui sont en gras sont ceux qui ont été utilisés lors de la phase d'entraînement pour l'identification des scripts. 102
7.2 Les résultats des algorithmes de discrimination entraînée avec les caractéristiques issues de l'histogramme de gradients orientés. Nous avons mené un ensemble de 10 tests sur les données jamais vues auparavant par les algorithmes de discrimination. ANN est l'abréviation pour le réseau de neurones artificiels, RF est l'abréviation des forêts aléatoires, et SVM est l'algorithme de machine à vecteurs de support. 119

7.3 Les résultats des algorithmes de discrimination de scripts entraînée avec les caractéristiques extraites avec l'auto-encodeur convolutif empilé. Les meilleurs résultats sont mis en gras. 124
7.4 Les résultats des algorithmes de discrimination de scripts entraînés avec les caractéristiques extraites par l'auto-encodeur convolutif empilé. Les meilleurs résultats sont mis en gras. 125
7.5 Les résultats des algorithmes de discrimination de scripts entraînée avec les caractéristiques extraites avec l'auto-encodeur convolutif empilé. Les meilleurs résultats sont mis en gras. 126
7.6 Les différentes initialisations de paramètres, pour la même architecture. 136

Table des figures

1 Example of the four scripts used for the image-based classification study in the second part of the thesis (a) Ethiopian (b) Syriac (c) Greek (d) Latin 5
2 Different forms of artifacts and degradation due to human intervention or the storage conditions (a) Black cover page (b) Stains and ink dillutions (c) Artifacts. (d) Crops made by human. 6

2.1 Un exemple d'un modèle graphique probabiliste modélisant la génération d'un document y par le choix d'un thème z. 21
2.2 (a) Exemple d'une page d'un manuscrit ancien écrit dans le script éthiopien. (b) L'interprétation numérique de l'image par l'ordinateur. La matrice des valeurs correspond à l'intensité des pixels dans le petit cadre en rouge de l'image (a). 22
2.3 Exemple des différentes variations manuscrites du chiffre 3 issu du jeu de données MNIST 22

3.1 La proximité sémantique est calculée entre les vecteurs représentant des mots uniques ou termes t dans un espace sémantique. 30
3.2 (a) Modèle graphique probabiliste à variables latentes. (b) Modèle graphique probabiliste hiérarchique à variables latentes. 32
3.3 Illustration du réseau bayésien de l'allocation latente de Dirichlet, les variables observées sont en gris, ici les mots $w_{d,n}$. 34
3.4 (a) Distribution presque uniforme de Dirichlet avec $\boldsymbol{\alpha} = \{0.99, 0.99, 0.99\}$. (b) Distribution symétrique de Dirichlet quand $\boldsymbol{\alpha} \to \infty$ avec $\boldsymbol{\alpha} = \{50, 50, 50\}$. (c) Distribution asymétrique de Dirichlet $\boldsymbol{\alpha} = \{15, 2, 5\}$ 34
3.5 Modélisation probabiliste d'un texte avec les différentes distributions des mots dans les thèmes à gauche de la figure et la modélisation de leurs proportions respective dans chaque document à droite de la figure. Illustration de [Blei 2012] 39
3.6 Définition d'une fonction d'adhérence sur les points d'un ensemble E 41
3.7 Illustration du principe de la fermeture d'un ensemble 43

4.1 Illustration de l'application de la fonction d'un fermé élémentaire F_x pour obtenir les fermés minimaux $\mathcal{F}_m = F_{mj}, j = 1, \ldots, k$ 53
4.2 Illustration du réseau des 133 utilisateurs avec deux types de relations qui les lient, la première est basée sur la distance de Hellinger ($distance \leq 0.15$) et les thèmes les plus discutés (avec une probabilité ≥ 0.15) 55
4.3 Le nombre d'éléments dans les sous-ensembles de fermés minimaux avec différents seuils : p_0 pour R_{MTP} et d_0 pour R_{d_H} 59

4.4 Illustration de la mesure de similarité avec les mêmes centroïdes initiaux (a) et différents centroïdes initiaux. E, est la distance d'Euclide, H la distance de Hellinger, P est la distance de l'adhérence. 62
4.5 Illustration du réseau bayésien de l'allocation latente de Dirichlet, les variables observées sont en gris, ici les mots $w_{d,n}$. 67
4.6 Flux de travail de l'implémentation Spark. Chaque itération est exécutée dans l'espace de travail *Map* et *Reduce*. Le modèle LDA est copié sur les p processeurs, et le document de travail fourni en entrée est divisé en p ensemble de documents. Après l'application de la procédure *Reduce* les paramètres globaux n^{wj} et n^{dj} sont diffuser à l'ensemble des répliques de LDA. Le logarithme de la vraisemblance est calculé sur l'ensemble des documents tests après chaque itération. 69
4.7 Illustration de la valeur de la perplexité par rapport au nombre d'itérations dans la figure (a) et par rapport au nombre de thèmes dans la figure (b). Dans la figure (a) nous comparons notre version distribuée avec la version distribuée de MALLET [McCallum 2002]. Dans la figure (b) nous comparons la perplexité de notre version distribuée avec une version séquentielle. 73
4.8 Illustration du gain de vitesse (a) par rapport aux nombres de processeurs (b) Illustration du gain de vitesse en tenant compte du passage à l'échelle. 74

5.1 Nous avons une structure avec 3 niveaux, 2 catégories principales, les catégories étant détaillées du niveau 1 au niveau 3. 82
5.2 La moyenne du taux de précision des stratégies de classification A et B. 83

6.1 Nombre de tweets pour chaque catégorie. 90
6.2 (a) Matrice de confusion pour un SVM entraîné avec 3 grammes extraits d'un mélange du dialecte tunisien et de l'arabe moderne. (b) Matrice de confusion pour un SVM entraîné avec 1 gramme extrait du dialecte tunisien. (c) Matrice de confusion pour un SVM entraîné avec 1 gramme extrait du dialecte tunisien. 94
6.3 Relation entre les résultats de prédiction lors de la classification et la taille des tweets. (a) SVM entraîné sur les tweets TN. (b) SVM entraîné sur les tweets ASM. 95

7.1 Exemple des quatre scripts correspondant aux quatre classes identifiées (a) Éthiopien (b) Syriaque (c) Grec (d) Latin 100
7.2 Exemple de différentes formes de dégradation, artefacts, intervention humaine que l'on peut trouver dans les manuscrits étudiés. (a) Une page de garde noire. (b) Taches sur la page et dilution de l'encre. (c) Artefacts. (d) Coupures et interventions humaines. 101

7.3 (a) Extraction du signal au niveau du pixel de chaque image. (b) Application successive du masque $[-1, 0, 1]$ sur l'axe des x et $[-1, 0, 1]^T$ sur l'axe des y. 103
7.4 Exemple de l'application du masque $[-1, 0, 1]$ sur l'axe x de l'image (a), obtention des caractéristiques représentées dans l'image (b). Application du masque $[-1, 0, 1]^T$ et obtention des caractéristiques représentées dans l'image (c). 104
7.5 (a) Illustration d'une fonction convexe avec une seule solution représentée par un minimum global. (b) Illustration d'une fonction non convexe avec plusieurs solutions, dont un minimum global. 106
7.6 Exemple d'un réseau de neurones artificiel avec une couche d'entrée, deux couches cachées et une couche de sortie. Nous avons illustré des liens bi directionnels entre les noeuds pour illustrer le sens de la convolution et le sens de l'algorithme de rétropropagation. 107
7.7 Illustration des réponses non linéaires des fonctions d'activations *softsign* (a) et *than*(b) 108
7.8 Caractéristiques/Poids partagées 110
7.9 Un auto-encodeur avec une seule couche cachée 112
7.10 Détection des piétons avec des histogrammes de gradients orientés . 115
7.11 Des histogrammes de gradients orientés divisés en 8 *bins*. Chacun des trois histogrammes correspond à la partie supérieure, celle du milieu et la partie inférieure de l'image du script arabe, donné ici en exemple. La flèche indique la contribution d'une partie spécifique du tracé à l'histogramme [Saidani 2015] 116
7.12 Calcul de l'histogramme des gradients orientés sur une image d'une page d'un manuscrit écrit en latin. 117
7.13 Matrice de confusion des algorithmes de classification avec les caractéristiques extraites avec l'histogramme de gradient orienté. La figure (a) illustre les résultats de discrimination du ANN. La figure (b) illustre les résultats de discrimination des forêts aléatoires. Enfin, la figure (c) illustre les résultats de discrimination exécutés par SVM. Les meilleurs résultats sont mis en gras. 119
7.14 Exemple des informations capturer par les filtres, dessinés en rouge, obtenus avec la deuxième, filtre de taille 10×10, et troisième architecture, filtre de taille 28×28. 121
7.15 Un auto-encodeur avec une seule couche cachée 122
7.16 Exemples des caractéristiques apprises à partir des quatre scriptes (a) première architecture (b) deuxième architecture (c) troisième architecture 123
7.17 Matrice de confusion des algorithmes de classification avec les caractéristiques apprises extraites avec l'architecture 1 de SCAE. La figure (a) illustre les résultats de discrimination du ANN. La figure (b) illustre les résultats de discrimination des forêts aléatoires. Enfin, la figure (b) illustre les résultats de discrimination exécutés par SVM. 125

7.18 Matrices de confusions des algorithmes de classification avec les caractéristiques apprises extraites avec l'architecture 2 de SCAE. La figure (a) illustre les résultats de discrimination du ANN. La figure (b) illustre les résultats de discrimination des forêts aléatoires. Enfin, la figure (b) illustre les résultats de discrimination exécutés par SVM. 126
7.19 Matrice de confusion des algorithmes de classification avec les caractéristiques apprises extraites avec l'architecture 3 de SCAE. La figure (a) illustre les résultats de discrimination du ANN. La figure (b) illustre les résultats de discrimination des forêts aléatoires. Enfin, la figure (b) illustre les résultats de discrimination exécutés par SVM. 127
7.20 Visualisation des caractéristiques apprises par l'auto-encodeur. Chaque point correspond à une valeur dans le vecteur de caractéristiques apprises après la phase d'encodage dans les différentes architectures. La couleur bleue correspond au script Éthiopien, la couleur rouge au Syriaque, la couleur verte au Grec et enfin le jaune au Latin. 128
7.21 (a) Calcul de la direction du gradient à l'étape θ_{t+1}, Où $g(\theta_t) = \varepsilon\nabla f(\theta_t)$. (b) Illustration d'une fonction objectif $f(\theta)$. 134
7.22 Architecture d'un réseau convolutif avec des couches de *max pooling* 135
7.23 illustration de la sortie de la fonction de perte lors de l'entraînement et de la justesse des résultats calculés sur les données de test 136
7.24 Architecture d'un réseau convolutif avec des couches de *max pooling* 137

A.1 Diagramme qui décrit le processus de l'apprentissage statistique. Diagramme inspiré des cours du Prof. Yasser Abu Moustafa 146
A.2 $m_H(N) = 8$ avec $N = 3$ 150
A.3 Un exemple d'un hyperplan optimal qui sépare les données avec une marge maximale 152
A.4 Schématisation du compromis entre la variance et le bias. 155

B.1 Les quatre cas possibles du chemin à deux arcs relient X à Y passant par Z : (a) Effet causal indirect ; (b) Effet probable indirect ; (c) Cause commune ; (d) Effet commun 157
B.2 Densité de probabilité de l'*a prioi* beta 161
B.3 Modèle graphique probabiliste pour l'exemple de l'inférence des paramètres de la distribution gaussienne avec l'application de la méthode de de Bayes Variationelle. Le modèle est graphe orienté de produits. 165

Classification du texte numérique et numérisé. Approche fondée sur les algorithmes d'apprentissage automatique.

Résumé : Différentes disciplines des sciences humaines telles la philologie ou la paléographie font face à des tâches complexes, fastidieuses et consommatrices en temps pour l'examen des sources de données. La proposition d'approches computationnelles en humanités permet d'adresser les problématiques rencontrées telles que la lecture, l'analyse et l'archivage de façon systématique. Les modèles conceptuels élaborés reposent sur des algorithmes et ces derniers donnent lieu à des implémentations informatiques qui automatisent ces tâches fastidieuses.
La première partie de la thèse vise, d'une part, à établir la structuration thématique d'un corpus, en construisant des espaces sémantiques de grande dimension. D'autre part, elle vise au suivi dynamique des thématiques qui constitue un réel défi scientifique, notamment en raison du passage à l'échelle. Une contribution de cette thèse concerne la distribution de ces calculs sur une grille d'ordinateurs, avec des outils tels que Spark, Hadoop et MapReduce.
La seconde partie de la thèse traite de manière holistique la page d'un document numérisé sans aucune intervention préalable. Le but est d'apprendre automatiquement des représentations du trait de l'écriture ou du tracé d'un certain script (langue écrite) par rapport au tracé d'un autre script. Il faut dans ce cadre tenir compte de l'environnement où se trouve le tracé : image, artefact, bruits dus à la détérioration de la qualité du papier, etc. Notre approche propose un empilement de réseaux de neurones auto-encodeurs afin de fournir une représentation alternative des données reçues en entrée. Les résultats obtenus avec les représentations apprises par les réseaux de neurones sont comparables à ceux fabriqués par des experts. De plus, nous proposons un retour d'expériences autour de l'initialisation des paramètres d'apprentissage en vue de pistes d'améliorations futures.
Les contributions de ce travail concernent la conception d'algorithmes pour la classification des textes numériques et numérisés. La classification des textes numériques est opérée à l'aide d'un modèle basé sur des espaces sémantiques quant à la classification des textes numérisée, elle s'appuie sur le développement d'architectures de réseaux de neurones.

Mots clés : Modèle graphique probabiliste, systèmes d'apprentissage, recherche d'information, Apprentissage automatique, apprentissage automatique des caractéristiques, réseaux de neurones.

Extended abstract

With developments in information technology in general and the Internet in particular, we see an unprecedented increase in the production, storage and exchange of textual documents or documents images. For example, with the emerging social media and short texts or the digitization of historical document motivation by the aim of preserving the cultural heritage. In both cases, it is important to provide the user with relevant information during the search phase. We find ourselves here within the framework of the Knowledge Data Discovery (KDD) whose implications for the development of user-oriented IT tools is a great challenge [Cao 2015].

Classifying a set of documents is a standard problem addressed in machine learning and statistical natural language processing [Manning 2009]. The usually undertaken approach for classifying documents consists in the use of a measure of similarity between a set of features that represents each document. In this dissertation, a document comes in a form factor of a text file or an image of a handwritten text. Thus, we will refer to the addressed problem as text-based classification in the former and image-based classification in the latter. Accordingly, this thesis is divided into two parts.

Text-based classification (also known as text categorization) examines the computer-readable ASCII text and investigates linguistic properties to categorize the text. When considered as a machine learning problem, it is also called statistical NLP [Manning 2009]. In this task, a text is assigned to one or more predefined class labels (i.e category) through a specific process in which a classifier is built, trained on a set of features and then applied to label future incoming texts. Several Machine Learning algorithms have been applied to text classification (see [Aggarwal 2012a] for a survey) : Rocchio's Algorithm, N-Nearest Neighbors, Naive Bayes, Decision trees, Support Vector Machine (SVM).

Text-based features are typically extracted from the so-called word space model that uses distributional statistics to generate high-dimensional vector spaces. Each document is represented by a vector of word occurrences. The set of documents is represented by a high-dimensional sparse matrix. Given a set of keywords, one can use the angle between the vectors as a measure of similarity between the documents. Nevertheless, this approach suffers from the curse of dimensionality due to the sparsity of the matrix that represents the documents.

The addressed open issues in the text-based classification part are as following.

— The representation of a set of text documents in a semantic space omitting, therefore, the use of the high-dimensional sparse matrix.

— On the one hand, tracking the changes that could occur within the thematic structure of the processed collection of documents.

— On the other hand, monitoring the dynamic of themes or topics raise another issue related to the scale of the collection of texts. The classification algorithms could consume an important amount of time when dealing with a streaming collection of documents.

— The organization of the collection of documents according to the discovered

topics and the relations that relate them to each other.

To classify the document based on their semantic content, the topics need to be extracted. This can be done using one of the three following methods. (i) lsa [Landauer 1997] uses the Singular Value Decomposition methods to decompose high-dimensional sparse matrix to three matrices : one matrix that relates words to topics, another one that relates topics to documents and a diagonal matrix of singular value. (ii) Probabilistic lsa [Landauer 1997] is a probabilistic model that treats the data as a set of observations coming from a generative model. The generative model includes hidden variables representing the probability distribution of the words in the topics and the probability distribution of the topics in the words. (iii) Latent Dirichlet Allocation [Blei 2003] is a Bayesian extension of probabilistic lsa. It defines a complete generative model with a uniform prior and full Bayesian estimator. The common assumption behind the three methods is that there exists a semantic relationship between terms which have high probability within the same topic — a phenomenon that is rooted in the word co-occurrence patterns in the text and that can be used for information retrieval and knowledge discovery in databases.

We selected the Latent Dirichlet Allocation to be the core algorithm in the text-based classification part for two main reasons : First, it is a generative model that remove the high-dimensional matrix from the workflow and give us instead latent variables to work with. The second reason and compared to pLSA, LDA includes an a priori to work with.

The following research hypotheses in the text-based classification part are formulated as follows

— Extracting and using the topics as a set of features, in opposition to keywords, will help us track the changes in the collection more effectively.
— In order to track the topics in a collection of documents, we may build a set of neighborhoods. In each neighborhood, the documents are related to each other by their major topic and a set of possible major or hot topics.
— Combining the horsepower of several instances of the same algorithm where each instance compute a portion of the collection of documents then combine its results may solve the problem of time consumption and scaling.

The contributions in this part are as follows.

— An efficient workflow for text representation and topic extraction that includes a sequential parallel and semi-supervised version of the Latent Dirichlet Allocation implemented in Spark Map Reduce.
— A novel multi-criteria classification method that combines pretopology and topic modeling.
— This thesis conducts an extensive investigation on the role of the number of topics, and the role played by the centroids when the similarity between a set of clusters is computed.
— A novel method is proposed to annotate large archives of documents with a hierarchical thematic structure. A document is labeled through several categories from a broad level to a more detailed level. For example, from the category games then puzzles to sudoku.

— Since the resources for Arabic Natural Language Processing (NLP) is still limited. We propose a dataset for sentiment analysis applied on the Tunisian dialect. The Arabic dialects are different and the need for data is real. We propose a benchmark for the future work.

The outline of the text-based classification part is as follows.

In chapter 3, we present the conceptual approach that we used to tackle the problem of the text-based classification. This approach is based on the core method called the Latent Dirichlet Allocation (LDA). LDA is a generative probabilistic model by Blei et al. [Blei 2003] that allows us to discover the latent structure (i.e. the topic structure) of a set of documents. LDA gives us three latent variables after computing the posterior distribution of the model ; the topic assignment, the distribution of words in each topic and the distribution of the topics in each document. Having the distribution of topics in documents, the pretopology allows us to compute the elements of each cluster of documents using the pseudoclosure function which gives us the ability to follow the process step by step. The pseudoclosure function is applied to the distribution of the topic, computed by LDA, in each document. We use the pretopology for its ability to manipulate many binary relationships. This conceptual approach allows us first, to model a topic space and to cluster the documents according to their positions in the topic space.

Document collections evolve through time and the structure of the topics that they contain change as well : one major topic a moment t can become obsolete at $t + 1$. We need then a classification algorithm that is able to take into account the major topics as well as the ones to be. In Chapter 4, we propose a multi-criteria classification algorithm and a distributed version of the latent Dirichlet allocation to handle the scaling.

LDA is a non-supervised algorithm for topics modeling. The computed themes are represented by a distribution of words that have different probabilities. The higher the probability the more likely a word belongs to a given topic. The interpretation and labeling of these probabilities are manual and subjective. We therefore propose a semi-supervised implementation to lead the algorithm to automatically learn the precise category and therefore automatically label a distribution of words. We then trained a random forest classifier to categorize the document in the hierarchy starting from the broad categories to the most detailed ones. This approach is useful to automatically build an archive from a large collection of documents or for ontology learning from several textual sources. This work is presented in chapter 5.

With the growth in the use of social media platforms. Sentiment Analysis methods become more and more popular while for Classical Arabic there are many datasets available, most of the Arab countries' Dialects suffer from very limited resources for training an automated system. In chapter 6, we study Sentiment Analysis applied on the Tunisian Dialect and Modern Standard Arabic by providing a manually annotated dataset. On this dataset, we performed feature selection and trained 6 classifiers to provide a benchmark for future studies.

In the second part of the dissertation, we study the script identification task with a special focus on the extraction of the features from the images of several

(*a*) (*b*) (*c*) (*d*)

FIGURE 1 – Example of the four scripts used for the image-based classification study in the second part of the thesis (a) Ethiopian (b) Syriac (c) Greek (d) Latin

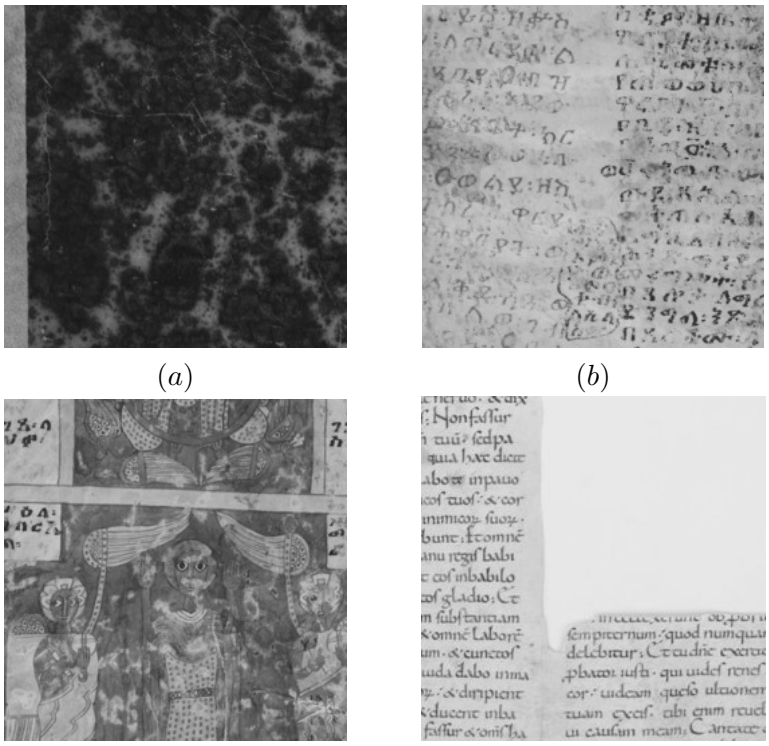
historical documents. Each image of the document is classified according to the script or written language. **Image-based classification** examines the image at a pixel level and investigates the properties of a hierarchy of patterns from the text blocks to the different artifacts found in the image of the document. The workflow of image-based classification method can be described in two main steps : First, extracting and selecting a collection of features. Then, feeding a classifier with those features.

The historical document images used in this part are downloaded from the National French Library [1]. The collected dataset is composed of 13 manuscripts for a total of 4769 pages divided into four principal scripts : Ethiopian, Syriac, Greek and Latin. Four examples of pages are presented in Figure 7.1. The images of pages contain several degradations due to both human (*e.g. physical intervention on the page with several crops*) and to the environment where the manuscripts were kept. Manuscripts written in Ethiopian come from London-Ciadailhet, Griaule and Marcel Cohen collections. Manuscripts written in Syriac comes from the Old Testament collections in it peshitta version. The pages of manuscripts written in Greek are from the chancellor Séguier collection. Finally, the one written in Latin comes from the collection of Gregoriuis Magnus, Homiliae in Ezechielem and roman lectionaries

1. http://www.bnf.fr/fr/collections_et_services/catalogues.html

composed from 670-680.

More effective hand-crafted methods for script identification have been introduced recently by [He 2017]. The several novel discriminative features are proposed following the joint feature distribution principal (JFD). Textural-based and Grapheme based features such as CoLBP inspired by the co-occurence pattern distribution, Co Hing and QuadHing based on the original Hinge Kernel, Ink Context inspired by the junction feature and shape context. The authors then fed a K nearest neighbor classifier to perform the script identification.



(*a*) (*b*) (*c*) (*d*)

FIGURE 2 – Different forms of artifacts and degradation due to human intervention or the storage conditions (a) Black cover page (b) Stains and ink dillutions (c) Artifacts. (d) Crops made by human.

It is clear the set of hand-crafted features are still useful for the script identification specially on the case of handwritten documents. Nevertheless, we believe that the historical document are a complex entity and the script identification task should take into account the environment (*i.e.* artifacts, degradation traces, see figure 2) of the handwritten text to be as solid as possible when the document corpora change or grow. These issues are emphasized by the lack of common standards for manuscripts digitization, or when we encounter fragmented manuscripts that are distributed across different libraries. These challenges demand flexible approaches that do not depend on several steps of pre-processing or controlled environment [Kang 2014].

With winning several contests [Assael 2016, Krizhevsky 2012] in image classification and pattern recognition **deep learning** becomes the recent research trend. In the Document Image Analysis context it has proven its adaptability for several tasks like layout analysis, writer identification, word spotting, etc. In the second part of the thesis, we are interested in learning the representations for the task of script identification. We believe that the convolutional auto-encoders have great potential to solve the problem of unsupervised learning of useful features without the need for labels.

Two research hypotheses are explored in this part :

— Due to the complexity of historical documents we need a holistic approach to extract the features of the scripts. When we extract the features we believe that taking into account all the layout of the page is as important as learning the different curvatures or gradients of the writing drawings.

— The learned abstract features don't have to be of the kind that label supervised learning induces in such a problem like the human writing.

The contributions of this part are as follows :

— We built a baseline for an unexplored set of manuscripts using both handcrafted features and a set of features learned with Convolutional Auto-encoders (CAE) and Convolutional Neural Networks (CNN).

— We propose a set of experiments to deepen our understanding of the behavior of CAE and CNN regarding different random initialization functions.

The image-based classification part is presented through the outline of Chapter 7. In this chapter we studied the script identification, the historical documents are classified according to the script or the written language that they contain in their text blocks. In the first section we describe our approach of feature extraction on the page level without any pre-processing (*i.e.* taking into account all the artifacts and different images). We then present the dataset of manuscripts that we have collected for our study and give some statistics. After a brief introduction in the second section of deep learning concepts and of pre-training methods used in the chapter, we build in the third section a baseline with the three following classifiers : a neural network with one layer, the supported vector machine and a Random Forest. The classifiers were fed with handcrafted features extracted on the page level with the Histogram of Oriented Gradient descriptors. After building the baseline, we performed a series of experiments with the convolutional auto-encoder (CAE) to deepen our understanding of the learned features that we can obtain from our dataset. We then studied the hypothesis of stacking several to obtain fine-grained features that may increase the classification accuracy. The obtained features were fed to the same classifiers used in the baseline. Our work also explored the features extracted from a convolutional neural network and allowed to reveal the importance of the function used for random initialization and the momentum parameter. We finally conclude and discuss satisfactory results of our approach and compare the learned and the handcrafted features.

Chapitre 1

Introduction

Avec les développements de l'informatique en général et de l'internet en particulier, nous constatons autour de la connaissance humaine une augmentation sans précédent de la production, du stockage et de l'échange de documents textuels ou d'images de documents textuels. Il est important de fournir à l'utilisateur l'information pertinente lors de la phase de recherche. On se retrouve là dans le cadre du Knowledge Data Discovery (KDD) dont les implications pour le développement d'outils informatiques orientés vers l'utilisateur constituent un grand défi [Cao 2015].

Nous proposons dans cette thèse des méthodes destinées aux humanités computationnelles. Les contributions faites se positionnent selon deux axes : le premier est celui de la fouille de textes, avec une application sur les textes échangés dans les réseaux sociaux. Le deuxième est celui de l'analyse des images des documents, avec une application sur les manuscrits anciens. Les deux axes sont présentés dans les deux parties de la thèse.

Dans la première partie, chapitre 3, 4 et 5, nous traitons des opérations de fouilles textuelles approfondies sur des espaces sémantiques. Les espaces sémantiques constituent une famille de méthodes mathématiques. Ils permettent de représenter la similarité sémantique de mots, de documents ou de concepts à partir de l'analyse des co-occurrences de termes qui sont établies à partir de grands corpus constitués de plusieurs dizaines de milliers de documents. Une propriété importante des espaces sémantiques est la métaphore topologique : les mots proches dans l'espace sémantique sont proches d'un point de vue sémantique.

Les applications de fouilles de textes utilisant des espaces sémantiques reposent sur une interprétation statistique du nombre d'occurrences d'un mot ou un groupe de mots au sein d'un document. Les co-occurences des mots par rapport au document peuvent être représentées par des matrices ou par des graphes. Par exemple, dans le cas d'une matrice, l'ensemble des mots sera représenté au niveau des colonnes, et quant aux documents ils seront représentés au niveau des lignes.

Les mots appartiennent à un vocabulaire d'une taille limitée alors que nous pouvons avoir un nombre important de documents au sein du corpus. Cela a pour résultat une représentation creuse avec un large écart dans ses dimensions (e.g. nombre étendu de lignes par rapport au nombre limité de colonnes). Afin d'éviter des calculs prohibitifs sur ce type de représentations, nous avons décomposé ces dernières en des composantes latentes, c'est-à-dire avec des profils qui résument au mieux les informations se trouvant dans la matrice de co-occurrences.

Les composantes latentes extraites de la matrice de co-occurrences correspondent

aux distributions des thèmes dans les documents et des mots dans les thèmes. Pour procéder à cette décomposition, nous avons utilisé la méthode de l'Allocation Latente de Dirichlet (LDA pour Latent Dirichlet Allocation) [Blei 2003]. LDA est un modèle probabiliste conçu pour l'identification des thèmes dans les textes. LDA modélise, par un mélange de distributions multinomiales, la production ou l'architecture cognitive en jeu pour l'écriture d'un texte. Le lien entre la cognition et le traitement automatique du language naturel est très courant en informatique [Poibeau 2016]. Le processus d'écriture dans LDA est décrit comme génératif, il est présenté comme suit : un auteur au moment d'écrire son texte choisit d'abord un ensemble de thèmes avec différentes proportions. Puis, pour chaque thème, l'auteur choisit différentes proportions de mots. L'auteur rassemblera enfin les mots choisis avec différentes combinaisons pour construire le texte de son document. Donc, dans ce modèle, chaque document est une distribution de thèmes et chaque thème est une distribution de mots.

En pratique, nous ne pouvons observer que les mots transcrits numériquement dans une matrice à travers leurs co-occurrences. Le processus génératif décrit précédemment est donc inversé : nous partons d'abord des valeurs de co-occurrences transcrites dans la matrice, nous y appliquons ensuite LDA pour arriver enfin aux représentations thématiques des documents. À travers l'utilisation de LDA, nous illustrons les représentations thématiques par des vecteurs de probabilités qui sont rassemblées dans un espace sémantique.

Nous avons proposé une analyse de l'espace sémantique par la structuration de ce dernier en des ensembles de documents. Cette approche ensembliste nous a permis de proposer deux nouvelles méthodes de classification des textes. La première, appelée classification multicritères, est fondée sur l'analyse structurale de l'espace sémantique latent créée par LDA. Chaque document dans cet espace est un vecteur de probabilités qui représente une distribution de thème. L'algorithme classifie un document suivant deux critères : un critère qualitatif qui se traduit par le thème majeur du document, et un critère quantitatif qui est la mesure de distance entre deux vecteurs de probabilités. Chaque vecteur, rappelons-le, représente la distribution des thèmes dans deux documents. La deuxième méthode, appelée classification hiérarchique, est fondée sur le développement d'une version semi-supervisée de LDA et l'utilisation des forêts aléatoires. Cette méthode propose de classifier un document à travers une hiérarchie de catégories partant de la plus générale vers la plus particulière.

Nous faisons l'hypothèse que la prise en compte des mouvements des vecteurs au sein de l'espace sémantique est d'une part réalisable. Et d'autre part, elle présente un intérêt dans de nombreuses applications, telles que la détection d'informations importantes dans l'actualité, la détection des modifications des thématiques abordées dans les documents générés en grand nombre telles que la production d'articles scientifiques, etc. Ce travail est présenté dans le chapitre 4.

Nous envisageons l'observation des mouvements au cours du temps au sein de l'espace sémantique par le biais de la prétopologie. La prétopologie est une théorie mathématique modélisant le concept de proximité et s'applique entre autres à

la modélisation dynamique de propagation, à la classification des données et à la reconnaissance des formes.

Nous exploitons la prétopologie du fait de sa souplesse qui nous assure un suivi, pas à pas, de processus de description et de construction des ensembles et aussi du fait de sa stabilité à l'égard des différent opérateurs mathématiques qu'elle utilise comme la connexité ou la continuité. La prétopologie est considérée pour appréhender les phénomènes dynamiques au sein des espaces sémantiques.

Le projet de thèse vise d'une part à construire des espaces sémantiques de très grande dimension, à identifier des structures thématiques latentes dans l'espace, et à les visualiser sous forme de réseau complexe à grande échelle. D'autre part, le projet vise à faire le suivi dynamique de ces structures thématiques. Le suivi dynamique constitue un défi scientifique, notamment en raison du passage à l'échelle liée à l'évolution de l'espace sémantique au cours du temps. Nous appréhendons ce problème par la conception de solutions de calcul parallèle. Les solutions proposées sont conçues à l'aide des systèmes multi-agents. Nous représentons chaque agent par un processeur qui calcule sur une portion du corpus la structure thématique. Chaque agent envoie la portion de ses résultats à un autre agent qui collecte les résultats et les unifie pour obtenir une seule sortie. Ce travail est présenté dans le chapitre 4 avec des outils comme Spark, Hadoop et MapReduce.

L'objet d'étude de la première partie de la thèse est le texte numérique. Nous qualifions un texte comme étant numérique lorsqu'il est saisi sur un ordinateur. L'objet d'étude de la deuxième partie de la thèse est le texte numérisé. Un texte est numérisé lorsqu'il se trouve dans une image stockée dans un ordinateur. Par comparaison à la première partie, la représentation du texte numérisé peut être approchée par des matrices ou des graphes. Par exemple, la matrice sera constituée de différentes valeurs, dans chacune de ses cellules. Ces valeurs correspondent à différentes intensités de pixels de l'image que nous souhaitons traiter.

Dans la première partie, nous avons déployé des algorithmes pour extraire des composantes à partir de chaque représentation des relations de co-occurrences mots-documents. Dans du texte numérique, il s'agit de composantes qui correspondent à des caractéristiques sémantiques. Dans la deuxième partie, et dans le cas du texte numérisé, il s'agit de composantes qui correspondent à des caractéristiques du tracé, par exemple, la largeur du trait, le contraste des pixels, etc.

Dans le chapitre 7 de la deuxième partie de la thèse, nous traitons la problématique de la discrimination des scripts associés aux documents anciens. Un script est une langue écrite. Le défi lié à cette tâche consiste à identifier automatiquement les caractéristiques du tracé des différents scripts. L'extraction automatique de ce type de caractéristiques en vue de l'entraînement d'un algorithme de classification reste à l'heure actuelle une tâche complexe. Une caractéristique doit être la plus représentative possible de l'ensemble des données en entrée. Par exemple, une caractéristique doit décrire au mieux dans un format compressé la matrice de pixels de chaque image. Une caractéristique doit être également la plus générale possible, afin que l'algorithme d'apprentissage qui l'utilise puisse généraliser son apprentissage sur d'autres jeux de données.

Lorsqu'on est en présence d'un manuscrit ancien sous forme d'un document numérisé, il s'agit généralement d'une image. Les caractéristiques sont accessibles au travers du traitement de l'image comme un signal que l'on transforme en utilisant des filtres. Un filtre peut être considéré comme un prisme qui décomposera le signal, représenté par une matrice de pixels, en un ensemble de caractéristiques. En pratique, le signal est une matrice de pixels et le filtre est une fonction mathématique qui renvoie une autre matrice de pixels correspondante aux caractéristiques.

Notre approche consiste à aborder le problème de l'identification des scripts en prenant le signal de l'image du document dans sa globalité. Nous appliquons des algorithmes pour extraire les caractéristiques au niveau de l'image de la page entière. Nous pensons que l'application du filtre au niveau de toute la page réduirait la difficulté de réutilisation des algorithmes développés, car, étant plus résistante aux bruits, elle peut être généralisée sur différents manuscrits. Nous distinguons deux sortes de filtres, les filtres faits main (traduction de *hand crafted features* en anglais) et les filtres appris (traduction de *learned features* en anglais). Les premiers sont conçus par des experts afin d'extraire des caractéristiques déterminées : par exemple la moyenne des gradients du tracé, le contraste des pixels, etc.

Comme filtre fait main, nous avons choisi la méthode des gradients orientés en partant de l'hypothèse que la dynamique de l'écriture à la main pourrait être capturé par la direction des gradients (HOG pour Histogram Oriented Gradient) [Dalal 2005]. HOG caractérise une image par une distribution de gradients. Chaque gradient représente un changement de direction dans l'intensité du pixel ou la couleur dans une image. Les caractéristiques HOG sont représentées par un histogramme qui compte l'orientation du gradient des pixels d'une image donnée.

Par opposition à la première catégorie de filtres, ceux qui sont développés au cours de l'exécution d'un algorithme d'apprentissage sont considérés comme des filtres appris. Les avancées technologiques et conceptuelles liées au domaine de l'apprentissage supervisé et non supervisé nous fournissent des outils intéressants pour entreprendre ce genre de tâche.

Par exemple, l'utilisation des réseaux de neurones et de plusieurs de leurs variantes spécialisées est adaptée aux problématiques liées à l'analyse des images de documents. Ces variantes, comme les réseaux de neurones à convolutions (noté CNN pour Convolutional Neural Network) [Lecun 1998a], ainsi que les auto-encodeurs (noté CAE pour Convolutional Auto-Encoders) [Ranzato 2007, Bengio 2007a], ont permis d'obtenir des résultats intéressants. Récemment, un réseau de neurones à convolutions appelé lipnet [Assael 2016] a dépassé le taux de justesse des experts lors de la lecture sur les lèvres.

Deux sous-catégories des filtres appris ont été considérées dans nos travaux. Les premiers sont appris de façon supervisée avec l'utilisation des CNN. Les CNN appliquent une suite de transformations non linéaires sur les données pour faire émerger des caractéristiques utiles à la classification. L'algorithme de rétropropagation est appliqué pour ajuster les caractéristiques suivant le résultat obtenu lors de la classification.

Par opposition aux filtres appris de façon supervisée, les CAE nous permettent

d'obtenir des filtres de façon non supervisée. Les CAE sont des réseaux de neurones qui sont entraînés pour fournir une représentation alternative des données d'entrée qu'ils reçoivent. En pratique, cela est effectué par l'application d'une suite de convolutions pour obtenir la représentation alternative puis d'une suite de dé-convolutions pour vérifier si la représentation obtenue est caractéristique des données en entrée fournies au réseau.

Nous faisons l'hypothèse qu'un apprentissage effectif des caractéristiques utiles à la classification est possible dans un premier temps. Et, dans un second temps, l'extraction de caractéristiques fines est abordable par un empilement de CAE afin de fournir une représentation alternative des données reçues en entrée. Ces représentations vont des plus générales dans les premières couches, par exemple le changement de contraste dans l'image, vers les plus particulières, par exemple le tracé d'un scribe (écrivain) par rapport à un autre. Les résultats obtenus avec les représentations CAE sont comparables à ceux fabriqués par des experts.

La suite de transformations non linéaires des données en entrées peut devenir rapidement complexe et difficile à représenter dans une équation. Nous avons décrit les réseaux de neurones avec des structures graphiques. Nous nous référons à la structure où sont disposées des couches de convolutions et les couches de classifications comme une architecture. Nous nous sommes référé à l'architecture comme étant profonde, une architecture avec plusieurs couches de convolutions ou couches de CAE [Bengio 2007b].

Dans la deuxième partie, nous avons déployé un ensemble d'algorithmes pour l'extraction des caractéristiques utiles à la classification des manuscrits anciens selon les scripts qu'ils contiennent. Notre hypothèse de travail était de considérer l'environnement du script dans sa totalité, c'est-à-dire au sein de la page entière. Cet environnement est caractéristique à chaque script et comporte donc une information globale qui est importante lors de la classification. Notre approche consistait à empiler des réseaux de neurones à convolutions de type auto-encodeurs. Les auto-encodeurs permettent d'apprendre des représentations globales dans les premières couches de la pile jusqu'aux représentations particulières dans les dernières couches de la pile. Ces représentations fines mélangent les propriétés du script aux propriétés de la page où ils se trouvent. Nous avons rapporté les résultats obtenus avec les caractéristiques apprises, et ce en les comparant aux caractéristiques extraites à l'aide d'une expertise humaine. Nous avons enfin relevé l'importance de l'initialisation des paramètres lors de l'apprentissage.

Cette thèse se positionne dans le cadre des algorithmes d'apprentissage automatique. Les méthodes présentées dans les deux parties répondent à la problématique de la production numérique et de la numérisation massive de toute sorte de documents. Nous nous trouvons dans le cadre des humanités computationnelles, où il s'agit d'approcher le traitement de l'information culturelle avec des méthodes algorithmiques. En particulier, les algorithmes développés serviront à extraire et comprendre des phénomènes intéressants comme la relation thématique entre les textes ou l'intervention de plusieurs scribes sur un texte ancien.

Chapitre 2

Contexte

Ce chapitre est l'objet d'une présentation générale de deux aspects essentiels à nos travaux de thèse : l'apprentissage automatique par la conception d'algorithmes et l'apprentissage contextualisé pour traiter des textes. Ce chapitre est composé de rappels des notions essentielles à la compréhension des contributions sur le sujet de nos recherches. Ce dernier porte sur l'analyse des textes dans le cadre des humanités computationnelles. Les textes traités sont soit saisis sur un ordinateur, soit manuscrits et numérisés sous la forme d'une image.

2.1 Notion d'algorithme d'apprentissage : le cadre théorique

Nous proposons, dans cette partie, une définition de la notion d'algorithme d'apprentissage en ayant recours aux travaux de [Gold 1967], [Valiant 1984], [Kearns 1994] et [Vapnik 1998].

D'abord, nous définissons un algorithme comme étant une machine de Turing qui s'arrête. Une machine de Turing possède un ruban divisé en un nombre infini de cases, une tête de lecture et une table de transition qui dicte le mouvement transversal de la tête de lecture. Une case du ruban peut être vide comme elle peut contenir soit une opération soit une variable. La table de transition comprend des opérations de lecture, d'écriture et de mouvement de la tête de lecture. Une machine de Turing qui tourne est une machine avec une tête de lecture active. Une machine qui s'arrête est une machine de Turing avec une tête de lecture immobile et qui a écrit un résultat.

Nous nous référons à un programme comme étant la table de transition d'une machine de Turing. Face à des problèmes dont la suite d'instructions est connue, nous pouvons écrire un programme de bout en bout. En revanche, face à des problèmes que l'on qualifiera de complexes, car ils font appel à des interactions entre plusieurs entités, la définition d'un programme est difficile. Par exemple, l'interaction des mots dans un texte produit un sens. L'analyse de ce sens est un problème que nous qualifions de complexe et que nous ne pouvons pas approcher avec un programme défini, écrit du début jusqu'à la fin. Ce type de problème nécessite une approche algorithmique avec la conception d'un apprenti.

Nous distinguons deux types d'apprentis :

1. un apprenti fort est appelé à se faire une représentation des données en observation de façon à être capable de générer des structures toujours correctes.

Cette approche a été initiée par l'article de [Gold 1967] qui s'est articulé sur deux aspects : l'étude de la structure linguistique et la possibilité d'acquérir cette structure.

2. Un apprenti faible, par comparaison, se fait une représentation des données en observation qui peut être incomplète pour des résultats qui ne sont pas toujours parfaits. Nous utiliserons dans cette thèse ce type d'apprentissage que l'on qualifie d'approximativement correcte et qui est défendu par [Valiant 1984] sous l'appellation PAC (pour *Probably Approximatively Correct*) (voir A.3).

Le cadre de travail de l'apprentissage approximativement correct est probabiliste. Étant donné un vecteur aléatoire X et un vecteur aléatoire Y, tous deux issus de la même distribution jointe $P(X, Y)$, nous admettons une fonction $f : X \to Y$ capable de prédire Y étant donné les valeurs en entrée dans X. Ce qui se traduit par une probabilité conditionnelle $p(Y|X)$. Dans ce cadre, la relation entre la paire (X, Y), c'est-à-dire le vecteur X en entrée et les variables aléatoires du vecteur Y en sortie passe uniquement par l'espérance conditionnelle $f(x) = E(Y|X = x)$. Cette relation est déterministe et peut s'écrire sous la forme $Y = f(X)$.

Dans le meilleur des cas, nous avons une probabilité de 1 pour une sortie qui correspond à la sortie Y. Dans le pire des cas, nous obtenons au contraire une probabilité de 1 pour une sortie qui ne correspond pas à la sortie Y. Cependant, en pratique, nous n'avons pas accès à la fonction de $f(X)$ capable de prédire la sortie Y étant donné le vecteur d'entrée X. Le principe de l'apprentissage sous PAC, consiste à approcher la fonction f avec un ensemble $\mathcal{H}$ de fonctions dénotées $\hat{f}$. Chaque fonction $\hat{f}$ est évaluée sur ses sorties Y. Une fonction $\hat{f}$ est retenue si le nombre de fois où nous obtenons une prédiction fausse de Y est petit.

L'évaluation de l'ensemble $\mathcal{H}$ de fonctions est plus longuement discuté dans les travaux de [Vapnik 1998] qui apporte une dimension statistique au problème de l'apprentissage. Vapnik et Chervonenkis introduisent une mesure de la capacité de l'ensemble $\mathcal{H}$ à classer les données X. Cette mesure est appelé dimension VC (pour Vapnik et Chervonenkis). L'apprentissage PAC ou l'apprentissage statistique de Vapnik approche le même problème, celui de la taille de l'échantillon des données X et la taille de l'ensemble $\mathcal{H}$ nécessaire pour inférer la fonction $\hat{f}$ idéale.

Les deux approches PAC et celle de Vapnik ont un lien presque filial avec une spécialité : l'inférence inductive où l'induction réfère à l'apprentissage d'un concept général à partir d'un ensemble d'exemples. Supposons que nous ayons maintenant un modèle probabiliste qui décrit une certaine structure, et quelques observations partielles de ce qui s'est réellement passé. La différence entre les deux modèles réside dans le fait que l'inférence inductive consiste donc à deviner la structure qui a généré les exemples observés, tandis que l'inférence statistique consiste à tirer des conclusions à partir des exemples observés.

2.2 Apprentissage contextualisé pour traiter des textes

Nous abordons la procédure d'apprentissage avec deux stratégies : une stratégie dite supervisée qui peut être approchée par des algorithmes de discrimination (la traduction de *classification* en anglais) ; une stratégie dite non supervisée qui peut être approchée par des algorithmes de classification (la traduction de *clustering* en anglais). Dans les deux stratégies, nous distinguons trois phases : la phase de l'entraînement ou de l'apprentissage, la phase de tests ou de l'évaluation de l'algorithme d'apprentissage avec des exemples non appris et enfin l'application sur des données nouvelles.

2.2.1 Apprentissage supervisé

Posons X comme étant l'espace en entrée et Y comme étant l'espace de sortie d'une fonction $f : X \to Y$. Considérons également $\mathcal{D}$ comme étant la distribution des données observées sur $X \times Y$. L'objet de l'*apprentissage supervisé* est de trouver, à partir d'un échantillon $S = (x_i, y_i)_{i=1}^{n} \sim \mathcal{D}^n$, une fonction f susceptible de reproduire Y ayant observé X avec une mesure d'erreur minimale sur un échantillon de données différent de S. L'échantillon en question est appelé échantillon de test. Nous définissons la mesure d'erreur sur l'échantillon test comme suit.

$$\text{Test}_{\mathcal{D}}(f) = \mathbf{E}_{(x,y)\sim\mathcal{D}}[L(f(x); y)] \tag{2.1}$$

où $L(z; y)$ est la fonction de perte définie sur $Y \times X$, bornée par une constante positive M. $L(z; y)$ calcule la perte à chaque fois que f renvoie z au lieu de renvoyer y. Ici, il est à noter que la mesure de l'erreur est additive. Dans d'autres cas, si elle est multiplicative, nous pouvons appliquer une transformation logarithmique qui nous ramènera de nouveau à une simple addition du nombre de fois où f s'est trompé dans ses prédictions.

Une fois que nous avons trouvé la valeur de $\text{Test}_{\mathcal{D}}(f)$ la plus petite possible, nous considérons que le problème de l'apprentissage est résolu. Nous utiliserons la fonction f pour prédire des sorties sur un nouveau jeu de données en entrée, qui n'a jamais été utilisé lors de l'entraînement, la capacité de f à prédire les bonnes sorties $\hat{y}$, est appelée capacité de généralisation.

Le cas idéal serait de trouver une fonction f avec un minimum global tel que

$$f = \arg\min \ \text{Test}_{\mathcal{D}}(f) \tag{2.2}$$

Cependant, cela est fondamentalement impossible à réaliser, car nous ne connaissons pas la distribution de probabilité $\mathcal{D}$. Nous chercherons, donc, la mesure d'erreur la plus petite possible sur l'échantillon S et généraliserons sur $\mathcal{D}$ l'apprentissage de f qui est issue d'un espace réduit de fonctions appelé aussi espace d'hypothèses $\mathcal{F}$. Nous définissons la mesure d'erreur sur l'échantillon d'entraînement comme suit.

$$\text{Entrainement}_S(f) = \mathbf{E}_{(x,y)\sim S}[L(f(x); y)] \tag{2.3}$$

Le problème illustré dans l'équation 2.2 est résolu par le choix de l'espace d'hypothèses $\mathcal{F}$ approprié (voir Annexe A section A.2.1).

$$f = \arg\min_{f \in \mathcal{F}} \ \text{Test}_{\mathcal{D}}(f) \tag{2.4}$$

La tâche de l'apprentissage dans le contexte supervisé consiste à minimiser la mesure d'erreur sur un échantillon d'apprentissage, ce qui fait de l'apprentissage supervisé une technique robuste et directement applicable sur des problèmes pratiques. Néanmoins, l'inconvénient de l'apprentissage supervisé réside dans le coût que représentent les données annotées ou préparées par l'être humain.

Configuration probabiliste. Notons que dans le cas où nous sommes confrontés à des données, en entrées, susceptibles de créer une certaine confusion, nous pouvons exprimer la sortie de la fonction d'approximation ou de prédiction $\hat{f}$ dans les termes d'une probabilité conditionnelle. Nous calculerons de ce fait la probabilité d'avoir en sortie une certaine valeur c de y étant donné une valeur de x issue de la distribution de données $\mathcal{D}$.

$$\hat{y} = f(x) = \arg\max_{f \in \mathcal{F}} p(y = c|x, \mathcal{D}) \tag{2.5}$$

Dans l'équation 2.5, nous calculons la sortie la plus probable pour x, appelée le maximum de l'*a posteriori* $p(y|x, \mathcal{D})$ connu sous le nom de MAP en anglais *Maximum A Posteriori*.

2.2.2 Apprentissage non supervisé

En l'absence d'une relation déterministe entre X et Y, ou en d'autres termes en l'absence de $\mathcal{D}$, l'objectif poursuivi par un algorithme d'apprentissage est autre. Il s'agit de chercher une structure qui regroupe les données de façon à ce qu'elles soient les plus semblables au sein d'un même groupe et les plus dissemblables entre elles pour deux groupes différents. C'est ce que nous qualifions de problème de classification (*clustering*).

Nous abordons le problème du sur-apprentissage, un problème commun aux algorithmes d'apprentissage supervisé et non supervisé.

2.2.3 Sur-apprentissage et sous-apprentissage

Nous reprenons l'analyse de [Sutskever 2013a], fondée sur les travaux de [Kearns 1994, Valiant 1984], qui soutient que le sur-apprentissage est représenté par la différence entre la mesure d'erreur $\text{Entrainement}_S(f)$ et la mesure d'erreur $\text{Test}_{\mathcal{D}}(f)$. Le compromis sur la taille de l'espace d'hypothèse $\mathcal{F}$, ou le compromis biais variance (expliqué dans l'annexe A.3.3), est décrit dans le théorème 2.1 au terme de la différence entre $\text{Test}_{\mathcal{D}}(f)$ et $\text{Entrainement}_S(f)$.

Théorème 2.1. *Si $\mathcal{F}$ est un espace fini et L_f une fonction de perte telle que $L_f : Y \times X \to [0, 1]$, alors le sur-apprentissage est uniformément borné avec une mesure de probabilité S tirée de l'échantillon d'apprentissage.*

$$\mathbb{P}_{S\sim\mathcal{D}^{|S|}}\left[Test_{\mathcal{D}}(f) - Entrainement_S(f) \leq \sqrt{\frac{log|\mathcal{F}| + log1/\delta}{|S|}}\ \forall f \in \mathcal{F}\right] > 1-\delta \tag{2.6}$$

Où δ, $0 < \delta < 1$ est le taux de fiabilité.

Les résultats illustrés dans l'équation 2.6 suggèrent que si la taille des données d'entraînement est plus grande que $\log|\mathcal{F}|$ alors la mesure d'erreur lors de l'entraînement a une valeur proche de la mesure d'erreur lors de la phase de test. Le terme $\log|\mathcal{F}|$ dans la borne illustrée dans l'équation 2.6, rapporte que chaque entraînement comporte un nombre constant de bits relatifs à la meilleure fonction dans $\mathcal{F}$.

Justification. Le théorème central limite fait que la mesure d'erreur $\text{Entrainement}_S(f)$ est centrée en $\text{Test}_{\mathcal{D}}(f)$ avec une queue de distribution gaussienne selon $1/\sqrt{|S|}$ (si $L_f : X \times Y \to [0,1]$). En conséquence, pour une fonction $f \in \mathcal{F}$, la probabilité que $\text{Entrainement}_S(f)$ dévie de $\text{Test}_{\mathcal{D}}(f)$ de plus de $|\mathcal{F}|/\sqrt{|S|}$ est exponentiellement petite dans $\log|\mathcal{F}|$. En d'autres termes, la probabilité que la différence entre la mesure d'erreur lors de l'entraînement et celle faite lors des tests soit plus grande que $\log|\mathcal{F}|$, pour les $|\mathcal{F}|$ fonctions, est arbitrairement petite.

Le théorème 2.1 n'assure pas néanmoins le succès de la phase d'entraînement, mais il souligne l'importance du choix de l'espace d'hypothèses $\mathcal{F}$ (voir annexe A), afin que l'une de ses fonctions $f \in \mathcal{F}$ ait un taux d'erreur le plus petit possible lors de la phase d'entraînement et que ce taux se vérifie, par la suite, lors de la phase de test.

La même analyse peut être appliquée sur le principe du sous-apprentissage à la différence que si l'on réduit la variance et donc la taille de $\mathcal{F}$, la valeur du biais devient large, et la différence entre $\text{Entrainement}_S(f)$ et $\text{Test}_{\mathcal{D}}(f)$ devient également grand (voir explication dans l'annexe A.2.1).

2.3 Mise en oeuvre de l'apprentissage

Étant donné un problème d'apprentissage tel qu'il est défini dans la sous-section 2.2.1, et en supposant que nous ayons trouvé une taille convenable de l'espace d'hypothèses $\mathcal{F}$ tel que c'est souligné par le théorème 2.1, alors le problème de l'apprentissage est résolu quand $\text{Entrainement}_S(f)$ converge en probabilité vers la valeur minimale de $\text{Test}_{\mathcal{D}}(f)$. Cependant, comme $\mathcal{D}$ est une distribution de probabilités qui nous est inconnue, la recherche du minimum global, comme décrit dans l'équation 2.2, ne peut être, pour l'instant, résolue que par une heuristique qui parcourt l'espace des hypothèses ou espace des échantillons jusqu'à ce qu'elle trouve un optimum, c'est-à-dire, suivant un échantillon optimal $\mathbf{S}$ et qu'elle trouve une valeur minimale de $\text{Entrainement}_{\mathbf{S}}(f)$ pour $f \in \mathcal{F}$.

Cette heuristique est ce que l'on appelle un "algorithme glouton" [Kearns 1994]. Par exemple, l'algorithme de descente de gradient [Lecun 1998b] est un algorithme glouton. Il adopte une stratégie qui suit une pente au bout de laquelle l'algorithme

calcule les valeurs d'un paramètre. Autrement dit, l'algorithme estime la valeur du minimum global de la fonction $f(\theta) \in \mathcal{F}$ où $\theta \in \mathbb{R}^{|\theta|}$ est l'ensemble de paramètres sur $\mathcal{F}$. Le problème d'optimisation est transporté de l'espace des hypothèses $\mathcal{F}$ à l'espace des paramètres $\mathbf{\Theta}$ où l'on cherche à trouver la valeur minimale de Entrainement$_{\mathbf{S}}(\theta)$.

2.4 Analyse de textes dans le cadre des humanités computationnelles

Nous décrivons, dans cette section, le sujet d'étude de la thèse qui est le texte et les stratégies d'étude à haut niveau de ce sujet. Nous entreprenons deux différents travaux sur cet objet en le considérant sous deux formes différentes ; à savoir le texte écrit en caractères ASCII/Unicode et le texte accessible en image ou numérisé. Nous nous intéresserons à la première forme à des fins d'analyse sémantique et à la deuxième forme à des fins d'identification de l'écriture ou du script.

2.4.1 Discrimination et classification des documents

La tâche de la classification ou de la discrimination de documents consiste à regrouper des documents en un ensemble de C classes suite à une interprétation statistique des fréquences de mots qui sont contenus dans le document numérique.

Dans le cas de l'apprentissage supervisé, cela revient à calculer la probabilité $p(y = c|x, \mathcal{D})$, où x est un vecteur de taille fixe. Les valeurs du vecteur x sont calculées suite à une représentation des mots comme un mélange d'objets contenu dans une collection (en anglais on se réfère à ce concept comme *bag of words*). Dans ce dernier contexte, nous ne tenons pas compte de l'ordre que les mots ont dans le document. La valeur d'une coordonnée x_{ij} du vecteur x est incrémentée de 1 à chaque fois que le mot i est présent dans le document j. Nous reviendrons en détail sur ce type de modélisation appelée "sac de mots" dans la section 3.1.1.

L'intérêt de cette approche est de placer la collection de documents dans une matrice de mots/documents où il y a une transition des caractères ASCII vers des valeurs numériques. La variabilité de ces différentes valeurs correspond à des "caractéristiques latentes" appelées aussi "caractéristiques cachées". Par exemple, lors de l'analyse sémantique d'une collection de documents, la variabilité des thèmes ou la variabilité des mots dans les thèmes peut correspondre à une caractéristique latente à travers laquelle nous pouvons analyser notre collection de documents.

Partons de l'hypothèse qu'un document y est écrit suite au choix d'une variabilité de mots et que cette variabilité corresponde à une caractéristique latente z qui représente un thème. Alors, si nous sommes amenés à construire un modèle pour décrire le processus d'écriture d'un document, nous supposerons que le thème z est la cause du document y et dénoterons ceci comme suit $z \rightarrow y$.

FIGURE 2.1 – Un exemple d'un modèle graphique probabiliste modélisant la génération d'un document y par le choix d'un thème z.

La figure 2.1 illustre le modèle graphique probabiliste (PGM pour *Probabilistic Graphical Model* en anglais, voir annexe section B.1) de la dépendance d'un document y par rapport au thème z. Le modèle dans la figure 2.1 est aussi appelé *modèle génératif*, car on dit de la variable z qu'elle génère la variable y.

Cependant, comme z est une structure latente ou cachée et que y représente un document observé (représenté en gris dans la figure 2.1), nous devons inverser la flèche et inférer la structure latente z par le calcul de la distribution *a posteriori* $p(z|y)$. Ce calcul est rendu possible par une inférence bayésienne où le problème est transféré d'un calcul effectif de z vers une estimation de z comme étant un paramètre (voir annexe section B.1.1).

La dernière distinction, à faire par rapport à notre modèle, concerne le nombre de paramètres ; s'il est fixé, alors nous avons un modèle paramétrique, s'il n'est pas fixé et qu'il augmente en fonction de l'échantillon d'entraînement S, alors nous avons un modèle non paramétrique.

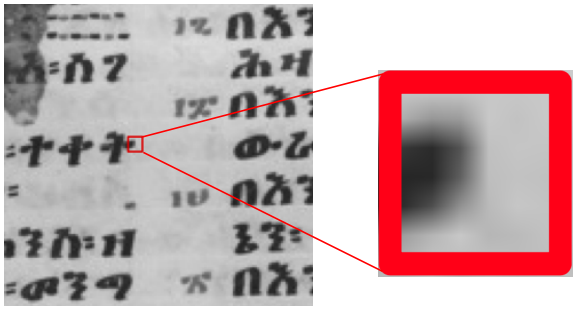
Face à des quantités importantes de données, les modèles paramétriques sont plus rapides à entraîner. Cependant, ils ont l'inconvénient d'imposer de fortes hypothèses sur la nature de la distribution des données $\mathcal{D}$. L'allocation latente de Dirichlet que nous utilisons dans nos travaux sur l'analyse sémantique est un modèle paramétrique. Les modèles non paramétriques ont l'avantage d'être plus flexibles, mais plus longs à entraîner et souvent incalculables en présence d'un jeu de données de taille importante. L'algorithme des K plus proches voisins est un exemple de ce dernier type de modèles.

Nous proposons, dans le chapitre 3, une étude fondée sur l'utilisation des modèles probabilistes paramétriques pour la construction d'un espace sémantique. Ils nous permettront de représenter la similarité sémantique de mots, de documents ou de concepts à partir de l'analyse de cooccurrences de termes établis à partir de grands corpus (plusieurs dizaines de milliers de documents). Les espaces sémantiques représentent les mots ou les documents sous la forme de vecteurs dans un espace vectoriel de grande dimension : les mots ayant un sens similaire sont représentés avec des vecteurs ayant des directions proches.

Afin de regrouper les textes des documents selon les thèmes qui les relient, nous nous intéresserons dans la section 3.3 à la métaphore topologique des espaces sémantiques où l'on considère que les mots proches au sein de l'espace représentent des mots proches du point de vue du sens. Nous introduisons les espaces prétopologiques afin de définir un concept de proximité sémantique entre les documents (voir

annexe section 3.3.1 pour une introduction à la prétopologie).

2.4.2 Analyse des images de textes



38	40	96	190
45	79	146	186
159	182	200	182
184	196	188	194

(*a*) (*b*)

FIGURE 2.2 – (a) Exemple d'une page d'un manuscrit ancien écrit dans le script éthiopien. (b) L'interprétation numérique de l'image par l'ordinateur. La matrice des valeurs correspond à l'intensité des pixels dans le petit cadre en rouge de l'image (a).

Le domaine de l'analyse des images des documents est à mi-chemin entre la reconnaissance de formes et l'analyse des images. Le sujet de la deuxième partie de la thèse porte sur la reconnaissance des scripts ou langues écrites. Les images de textes sont divisées en des petites régions qui peuvent être de différentes tailles, se trouvant à des coordonnées aléatoires sur l'image. Un exemple d'une de ces régions est représenté par le petit carré rouge illustré dans la figure (a) 2.2. L'interprétation numérique de la région qui correspond au carré rouge est illustrée dans la figure (b) 2.2, où les différentes valeurs correspondent à l'intensité des pixels de cette région.

Nous appliquerons une suite de transformations linéaires sur les valeurs numériques de la figure (b) 7.3 pour obtenir une autre suite de valeurs qui correspondra aux changements de la luminosité, du contraste ou d'autres caractéristiques. Nous utiliserons ces caractéristiques pour discriminer les régions qui contiennent un script c donné avec une forte probabilité présentée sous la forme suivante $p(y = c \mid x)$ où x est le vecteur de caractéristiques. Le défi consiste à trouver le vecteur de caractéristiques qui nous donne une représentation effective de l'image et un bon taux de justesse dans les résultats de discrimination.

FIGURE 2.3 – Exemple des différentes variations manuscrites du chiffre 3 issu du jeu de données MNIST

Un exemple de jeu de données classique qui est utilisé dans la littérature et connu sous le nom de MNIST (pour *Modified National Institute of Standards*). MNIST est un jeu de données constitué des dix chiffres manuscrits, avec 60,000 images

consacrées à la phase d'entraînement et 10,000 images gardées pour la phase de test. Ce dernier a été introduit dans les travaux de [Lecun 1998b] sur la reconnaissance de l'écriture. Dans le cas des algorithmes entrainés sur MNIST, il s'agit de reconnaître les chiffres. Dans la figure 2.3, nous avons illustré un exemple de quatre différentes écritures du chiffre 3.

MNIST est un jeu de données qui a été standardisé (i.e. les chiffres sont centrés, et la taille de l'image est la même, 28×28 px). Dans cette thèse, nous nous intéresserons à l'inverse des jeux de données brutes où il n'y a pas eu d'intervention humaine, par exemple les images des documents anciens.

Nous abordons en effet dans la deuxième partie de cette thèse la problématique de l'identification des scripts dans les manuscrits anciens. Nous testons les réseaux de neurones comme des outils nous permettant d'apprendre les caractéristiques du tracé des scripts pour mieux les identifier au sein d'une image d'un texte sans traitement préalable. En particulier, nous souhaitons exploiter les dernières avancées des réseaux de neurones pour apprendre de manière non supervisée une représentation alternative des données en entrées et exploiter cette représentation au sein de la tâche de l'identification des scripts.

Nous proposons au sein de cette thèse un ensemble d'algorithmes d'apprentissage supervisé et non supervisé pour l'approche des problèmes dans le contexte des humanités computationnelles comme celui du traitement des textes numériques et numérisés sous forme d'images de documents. L'objectif est de fournir *a posteriori* une palette d'outils aux chercheurs en humanités numériques.

Première partie

Analyse automatique des textes numériques pour la classification thématique

Chapitre 3

Structuration thématique des corpus de textes

Résumé : Nous abordons le problème de la classification des textes avec la méthode LDA, allocation latente de Dirichlet. Cette méthode due à [Blei 2003] est décrite par un mélange de distributions multinomiales qui correspondent à la distribution des mots dans les thèmes et la distribution des thèmes dans les documents. LDA est un modèle génératif : chaque distribution est reliée à une autre par un lien de parenté, l'ensemble est ainsi décrit dans un réseau bayésien. Dans ce réseau, chaque distribution est représentée par une variable aléatoire qui peut être latente ou observée. Les variables latentes sont calculées par une approximation fondée sur les variables observées. Ce modèle génératif nous permet d'éviter la manipulation de matrices creuses de grandes dimensions sur laquelle le temps d'exécution des algorithmes de classification serait prohibitif. La première étape du travail consiste à calculer les thèmes qui sont présents dans la collection de textes. Un thème est représenté par une variable latente dans LDA. Nous avons donc implémenté l'échantillonnage de Gibbs qui est un algorithme d'approximation. Notre implémentation permet de calculer la distribution des mots dans les thèmes et la distribution des thèmes dans les documents. La seconde étape consiste à rapprocher les documents selon les thèmes qu'ils contiennent afin de les classifier. Pour ce faire, nous avons besoin d'obtenir une représentation flexible et opérationnelle de la proximité, ce qui nous a ramenés à introduire la prétopologie. Nous faisons dans ce chapitre des rappels nécessaires et ciblés pour notre cadre de travail. En effet, un espace prétopologique pour les documents est construit en utilisant une collection de relations binaires où le calcul des "fermés" permet de structurer cet espace selon les thèmes qu'ils partagent.

Nous nous plaçons dans le cadre de la classification des documents. Les parties présentées correspondent à une introduction conceptuelle et méthodologique des travaux de classification de textes que nous allons aborder dans les chapitres 4 et 5. L'intérêt de ces travaux consiste à : (i) organiser des documents donnés pour faciliter la recherche d'informations, (ii) générer automatiquement des résumés (iii) catégoriser les documents dans une structure définie, par exemple une archive web ou une hiérarchie de catégories partant de la plus générale vers la plus particulière.

Afin de procéder à la classification, les textes sont représentés numériquement par les occurrences des mots qu'ils contiennent. La dimension de cette représentation est importante ; cependant, elle est creuse car un document ne peut contenir qu'un nombre réduit de mots qui sont tirés du lexique de toute la collection, nombre qui par exemple peut être de l'ordre de 10^7. Cette problématique est mise en exergue

dans le cas des textes provenant des échanges sur les réseaux sociaux. Par exemple, dans le cas de Twitter les documents ne dépassent pas 140 caractères.

Nous utilisons dans nos travaux une méthode probabiliste pour décomposer la représentation initiale de la collection de documents en un ensemble de variables latentes. Ces variables latentes peuvent être considérées comme des portraits-robots de la représentation initiale, ils nous permettent de répondre au problème de la dimensionnalité déjà évoqué. En pratique, ces portraits-robots ou variables latentes correspondent aux thèmes des différents textes. Ils nous permettent de construire un espace sémantique où les documents peuvent être rapprochés par leur proximité thématique à condition de définir une distance appropriée et un voisinage. Nous utilisons pour cela la prétoplogie un affaiblissement de la topologie. La prétopologie est un outil de modélisation du concept de proximité dans les espaces discrets [Belmandt 2011]. Dans notre cas c'est l'espace sémantique composé d'un ensemble de documents.

Ce chapitre est organisé comme suit. Dans la première section, nous positionnons la problématique de la classification de documents par rapport à l'état de l'art. Dans la seconde section, nous détaillons la méthode utilisée pour la réduction de la matrice en composantes latentes (i.e. décrit par interdépendance avec d'autres variables). Enfin, dans la troisième section, nous présentons notre méthodologie pour modéliser la notion de voisinage.

3.1 Le problème de la classification des textes

La catégorisation ou classification automatique des textes est l'investigation par ordinateur des propriétés linguistiques des caractères ASCII qui forment le contenu textuel d'un document numérique. Dans le cadre de l'apprentissage automatique, nous nous référons à la littérature du traitement statistique du langage "naturel" (*Natural Langage Processing*, NLP) [Manning 2008].

Le processus de classification d'une collection de textes consiste à étiqueter chaque texte avec une ou plusieurs classes (catégories) prédéfinies. Dans ce processus, un algorithme est d'abord conçu, puis il est entraîné avec un ensemble de caractéristiques spécifiques, par exemple les occurrences de mots ou encore les distributions de thèmes dans un document. Une fois entraîné, l'algorithme est utilisé pour étiqueter de nouveaux textes. Ces derniers sont différents des textes utilisés lors de l'entraînement. L'algorithme est évalué sur le nombre d'erreurs de classification obtenu lors de la phase d'apprentissage et lors de la phase de test (voir section 2.2).

Lors de l'entraînement de l'algorithme de classification, la phase d'extraction des caractéristiques utilisées pour l'apprentissage est cruciale. Les caractéristiques extraites à partir des textes sont typiquement issues d'un espace vectoriel de grande dimension. Cet espace est construit avec une modélisation vectorielle des mots à l'aide de la sémantique distributionnelle [Harris 1954].

Une fois les caractéristiques extraites, plusieurs algorithmes d'apprentissage (voir [Aggarwal 2012b]) peuvent être utilisés pour la classification automatique de textes,

pour citer les plus connus : algorithme de Rocchio's, méthode des K plus proches voisins, classification naïve bayésienne, les arbres de décision, les séparateurs à vastes marges (SVM), les forêts aléatoires.

Nous abordons ci-après la représentation numérique d'une collection de documents sous la forme d'une grande matrice creuse. En premier lieu, nous présenterons les méthodes qui utilisent l'espace vectoriel issu de la matrice. En second lieu, nous détaillerons les méthodes probabilistes utilisées qui nous évitent une interaction directe avec la matrice creuse de grandes dimensions.

3.1.1 Mathématisation des corpus de textes et modéles de représentation thématique

Nous abordons la mathématisation des corpus de textes avec la sémantique distributionnelle et à travers l'utilisation des matrices. Nous présentons ensuite les modèles de représentation des thèmes et justifions notre choix porté sur l'allocation latente de Dirichlet.

$$
\mathrm{M}_{DocsMots} = \begin{array}{c} \\ doc_1 \\ doc_2 \\ doc_3 \\ doc_4 \\ doc_5 \end{array}
\begin{array}{c}
\begin{array}{cccccccccc} mot_1 & mot_2 & mot_3 & mot_4 & mot_5 & mot_6 & mot_7 & mot_8 & mot_9 & mot_{10} \end{array} \\
\left(\begin{array}{cccccccccc}
1 & & & & 2 & 4 & & 3 & 1 & 1 \\
 & 1 & & 5 & & 1 & & 4 & 2 & \\
2 & 1 & 4 & & & 1 & & 1 & & \\
 & & 2 & 1 & 3 & & 1 & & & \\
4 & & 1 & & 2 & 1 & & 2 & & 1
\end{array}\right)
\end{array}
$$

Une collection de documents D est caractérisée par un nombre de mots N et un vocabulaire, ou liste de termes uniques, de taille V. Pour représenter de façon quantitative les mots dans les textes, nous utiliserons une représentation en histogramme du vocabulaire par rapport à la collection D, décrite dans une matrice de cooccurrence documents-mots (voir la matrice $M_{DocsMots}$ donnée en exemple). Dans cette représentation, chaque document correspond à un vecteur et chaque valeur dans le vecteur correspond au nombre d'occurrences de chaque terme unique.

Cependant, même pour des petites collections, nous avons un nombre conséquent de termes uniques ; premièrement cela nous donne des vecteurs d'une grande dimension. Deuxièmement, avec l'augmentation de la dimension ou la taille du vocabulaire V, les données deviennent plus clairsemées. Un document ne peut pas avoir dans son texte tous les termes du vocabulaire. Cela se traduit par une matrice creuse (beaucoup de zéros), où un nombre accessible et fini de documents ne peut pas couvrir tous les termes du vocabulaire.

Les deux caractéristiques énoncées soulèvent deux problèmes. D'abord, la conception d'algorithmes avec un pouvoir prédictif de qualité devient une tâche difficile, car le calcul de l'espérance $f(x) = E(Y|X = x)$ est affecté par la dimension de X (voir section 2.1 pour un rappel). Ensuite, la probabilité que deux documents tirés de la collection D soient proches est faible [Steinbach 2004] à cause de la matrice creuse.

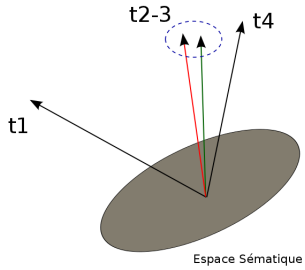


FIGURE 3.1 – La proximité sémantique est calculée entre les vecteurs représentant des mots uniques ou termes t dans un espace sémantique.

Une première application sur cet espace vectoriel consiste à utiliser des mots-clés pour identifier des vecteurs d'intérêt et calculer le cosinus de l'angle de ces vecteurs. Deux vecteurs identiques forment un angle nul et ont donc un cosinus d'une valeur égale à 1. Opposés, les deux vecteurs ont un cosinus égal à -1 et quand ils sont perpendiculaires ils ont un cosinus égal à 0. La similarité sémantique entre les documents varie entre -1 et 1.

$$\cos\theta = \frac{u \cdot v}{||u|| \times ||v||} \tag{3.1}$$

Le cosinus est calculé avec la fonction 3.1 où u et v sont deux vecteurs.

Cette approche, attirante par sa simplicité, mène souvent à des erreurs d'estimation de la similarité sémantique, car elle ne modélise qu'une part réduite de la variance des données. Par exemple, lors de la phase de recherche, une requête composée d'un mot ou d'un document est modélisée de façon symétrique.

Une autre approche consiste à classifier les documents en tenant compte de leur contenu sémantique. Dans ce cas, les thèmes sont à extraire. Pour ce faire trois méthodes se distinguent dans la littérature.

1. L'analyse sémantique latente (LSA) [Landauer 1997] utilise la méthode de décomposition en valeurs singulières (*Singular Value Decomposition*, SVD) pour décomposer la matrice creuse A d'une grande dimension en trois matrices.

$$A = U\Sigma V^T \tag{3.2}$$

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{bmatrix} \tag{3.3}$$

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \sigma_3 u_2 v_3^T \cdots \sigma_n u_n v_n^T \tag{3.4}$$

Dans l'équation 3.2, la matrice A est décomposée en une matrice U qui relie

les documents aux thèmes, une deuxième V^T qui relie les thèmes aux termes, et une troisième matrice Σ diagonale de valeurs singulières.

2. L'analyse sémantique latente probabiliste (pLSA) [Hofmann 1999] est l'extension probabiliste de LSA ; le modèle traite les données comme un ensemble d'observations issues d'un modèle génératif qui nous évite la manipulation de la matrice creuse à grande dimension.

   Le modèle génératif qui nous intéresse comprend des variables cachées qui correspondent à la distribution de probabilités des mots dans les thèmes et la distribution de probabilités des thèmes dans les documents.

3. L'allocation latente de Dirichlet (LDA) [Blei 2003] est une extension bayésienne de la version probabiliste de LSA. LDA définit un modèle génératif complet en incluant, contrairement à pLSA, des *a priori* et un raisonnement bayésien. Le raisonnement bayésien nous permet à son tour d'utiliser des algorithmes d'approximation pour calculer les thèmes dans la collection de textes.

Les trois méthodes sont reliées par l'hypothèse commune selon laquelle il existe une relation sémantique entre les termes (ou mots uniques du vocabulaire) qui possèdent une forte probabilité d'être présents dans le même thème. Autrement dit, un mot w dans un document d est fort probablement associé à un thème z si dans le même document il y a plusieurs mots qui sont affectés au thème z. Cette hypothèse trouve racine dans celle du distributionnalisme en linguistique [Harris 1954].

Ces patrons de cooccurrence peuvent être utilisés pour la recherche d'informations ou la découverte de nouvelles connaissances dans les bases de données.

Nous trouvons aussi dans la littérature l'utilisation des graphes pour représenter le texte en partant de plusieurs bases [Jiang 2010] : (i) les mots comme étant des unités ASCII (ii) les mots comme étant des unités grammaticales (iii) l'ordre des mots (iv) les hyponymes [1] des mots (v) la structure des phrases (vi) la division des phrases, et (vii) l'ordre des phrases. Une structure en graphe pour représenter le texte peut être générée à partir de ces propriétés où les arêtes (orientées ou non) modélisent les relations sémantiques entre les noeuds qui peuvent être des mots, phrases, ou textes entiers [Jiang 2010]. Une étude comparative avec les méthodes classiques a été présentée par Jiang et al. [Jiang 2010]. Les auteurs ont utilisé une version améliorée de l'algorithme *gSpan* [Yan 2002] afin de chercher des sous-graphes qui soient utilisés pour fournir des caractéristiques au moment de l'entraînement d'algorithmes de classification. Les sous-graphes sont similaires aux sous-matrices obtenues avec la décomposition en valeur singulière utilisée par LSA.

Nous utiliserons dans nos travaux la méthode de l'allocation latente de Dirichlet, car elle nous permet d'éviter la manipulation des matrices à grande dimension par la construction d'un modèle génératif représenté dans un réseau bayésien. Cela nous permet d'appliquer un raisonnement bayésien et d'utiliser des algorithmes probabilistes pour l'approximation de l'a posteriori.

1. Les hyponymes représentent une relation hiérarchique entre les mots, où chaque mot est l'extension sémantique de l'autre. Rapide est l'hyponyme de course

3.2 L'allocation latente de Dirichlet

Nous adopterons dans nos travaux sur l'analyse de la structure latente des textes un raisonnement bayésien (voir la section B.1.1) fondé sur l'identification des variables latentes et leur représentation compacte au sein d'un réseau bayésien. Les réseaux bayésiens sont ce que l'on appelle des modèles graphiques probabilistes.

3.2.1 Caractérisation de notre approche par les modèles graphiques probabilistes

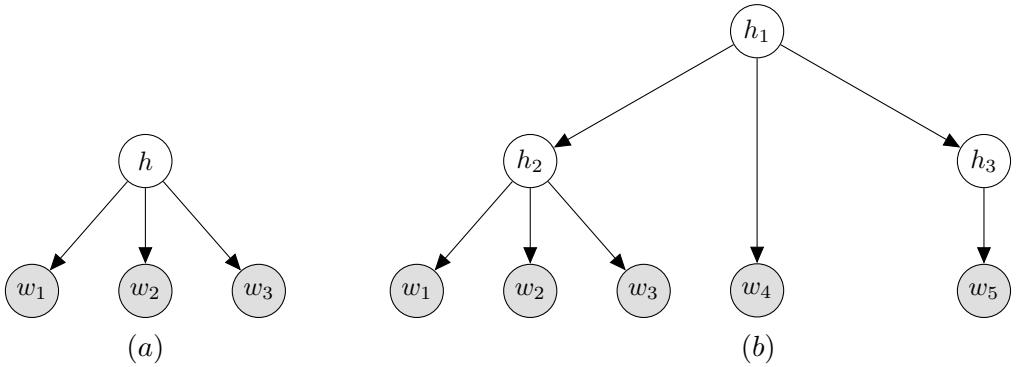


FIGURE 3.2 – (a) Modèle graphique probabiliste à variables latentes. (b) Modèle graphique probabiliste hiérarchique à variables latentes.

Les modèles graphiques probabilistes sont une représentation graphique des variables aléatoires et de leurs interdépendances. Les modèles graphiques à variables latentes [Koller 2009] sont utilisés dans le cadre de l'apprentissage automatique non supervisé où la classe que nous cherchons à mesurer est inconnue [Zhang 2004]. La classe inconnue est représentée par une variable cachée h (latente) illustrée dans la figure 3.2 (a), h ne faisant pas partie des variables observées $\mathbf{w}$, représentées en gris dans la figure 3.2.

La figure 3.2 (b) illustre une hiérarchie de variables latentes h_1, h_2, h_3 où une variable latente peut être conditionnée par une autre. Les différents niveaux de dépendances conditionnelles nous permettront de modéliser des structures complexes comme celle des thèmes dans les textes par exemple en rajoutant des connaissances *a priori* sous la forme de distribution. Nous pouvons aussi fixer des contraintes supplémentaires sous la forme de paramètres, paramètres conditionnant les variables latentes. Ces contraintes sont appelées des hyper-paramètres qui sont des paramètres que l'on fixe sur les paramètres du modèle d'apprentissage pour éviter notamment le surapprentissage.

Définition 3.1. Un modèle bayésien hiérarchique est un modèle statistique bayésien où la loi *a priori* est composée de distributions conditionnelles.

$$p(\theta|\theta_1), p(\theta_1|\theta_2), \ldots, p(\theta_{k-1}|\theta_k), p(\theta_k). \tag{3.5}$$

les paramètres θ_i, i = 1, ..., k, sont appelées des hyper-paramètres.

Les modèles graphiques probabilistes illustrés en exemples dans la figure 3.2 sont des réseaux bayésiens (voir la définition B.1). Nous retrouvons les modèles bayésiens hiérarchiques dans de nombreuses modélisations de structures qualifiées de complexes. Décrites par un réseau bayésien, ces structures nécessitent plusieurs variables et observations interdépendantes qui constituent une distribution jointe. Par exemple, la structure thématique dans les textes, décrite dans la distribution jointe, équation 3.6, issue du réseau bayésien (illustré dans la figure 3.3), a nécessité l'introduction de plusieurs niveaux de distributions *a priori*, de Dirichlet, sur les paramètres (appelés hyper-paramètres) constituant ainsi un modèle hiérarchique appelé l'allocation latente de Dirichlet.

$$p(\phi_k, \beta, \theta_d, \alpha, z_{d,n}, w_{d,n}) = \left(\prod_{k=1}^{K} p(\phi_k|\beta)\right)\left(\prod_{d=1}^{D} p(\theta_d|\alpha)\left(\prod_{n=1}^{N} p(z_{d,n})p(w_{d,n}|z_{d,n}, \phi_{1:K})\right)\right) \tag{3.6}$$

L'apprentissage des paramètres (différentes variables aléatoires latentes) des modèles graphiques probabilistes [Koller 2009] se fait par différentes méthodes comme l'algorithme de maximisation de l'espérance, les méthodes variationnelles (voir sous-section B.1.1) ou encore des algorithmes de simulation par chaîne de Markov avec des méthodes de Monte-Carlo (MCMC pour *Monte-Carlo Markov Chain*) ainsi que par l'algorithme de l'échantillonnage de Gibbs.

Nous utiliserons dans nos travaux l'algorithme de Gibbs pour sa capacité de simulation de distributions multidimensionnelles, ce qui fait de lui le candidat parfait pour calculer l'*a posteriori* d'un modèle bayésien hiérarchique. De plus, l'algorithme de Gibbs est un algorithme probabiliste par opposition par exemple à celui de la maximisation de l'espérance qui est déterministe. Cela revient à dire que nous adoptons une stratégie où nous calculons une solution approximativement correcte en un temps réduit plutôt que de calculer une solution exacte en un temps arbitrairement lent. Enfin, l'échantillonnage de Gibbs est adapté à notre approche qui est fondée sur le modèle de l'allocation latente de Dirichlet qui est également probabiliste.

3.2.2 Présentation de notre approche

L'allocation latente de Dirichlet (LDA) de [Blei 2003] est un modèle probabiliste génératif, construit pour des collections de groupes de données à caractère discret. Chaque groupe est considéré comme un mélange aléatoire d'un ensemble de thèmes dits latents. Chaque thème est une distribution de probabilités sur le vocabulaire de la collection de documents. LDA étant applicable sur toute collection de données à caractère discret, nous utiliserons le modèle dans le cadre du traitement automatique/statistique du langage naturel (NLP).

Nous nous référons à la collection de documents comme corpus et aux données comme des mots. LDA est un modèle de mélange de distributions dont l'hypothèse principale sur les données est que ces dernières sont générées à partir d'un mélange

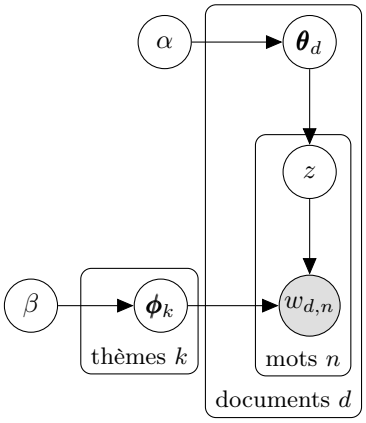
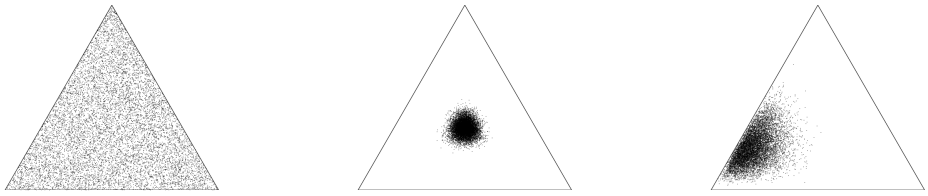


FIGURE 3.3 – Illustration du réseau bayésien de l'allocation latente de Dirichlet, les variables observées sont en gris, ici les mots $w_{d,n}$.

de distributions, et non pas d'une seule distribution.

Le modèle probabiliste génératif de LDA est utilisé pour l'apprentissage non supervisé des variables latentes qui sont, le thème z affecté au mot w, la distribution θ_d des thèmes z_d dans chaque document et enfin la distribution ϕ_k des mots dans chaque thème z. Ces variables aléatoires font partie d'un modèle génératif structuré avec les réseaux bayésiens.



(a) (b) (c)

FIGURE 3.4 – (a) Distribution presque uniforme de Dirichlet avec $\boldsymbol{\alpha} = \{0.99, 0.99, 0.99\}$. (b) Distribution symétrique de Dirichlet quand $\boldsymbol{\alpha} \to \infty$ avec $\boldsymbol{\alpha} = \{50, 50, 50\}$. (c) Distribution asymétrique de Dirichlet $\boldsymbol{\alpha} = \{15, 2, 5\}$

Le modèle génératif de LDA illustré dans la figure 3.3 décrit la distribution jointe de l'équation 3.6 qui est un mélange des dépendances conditionnelles entre les différentes distributions de probabilités qui sont θ_d et ϕ_k. Chaque document d a différentes proportions de thèmes qui sont représentées par la distribution θ_d. À chaque position dans le document d, un thème z est échantillonné à partir de θ_d. Enfin un mot est choisi en fonction de la distribution des mots ϕ_k dans le thème z. Les deux distributions θ_d et ϕ_k sont représentées par deux distributions de Dirichlet avec α et β comme hyper-paramètres.

Définition 3.2 (Distribution de Dirichlet)**.** Soient $\alpha_1, \alpha_2, ..., \alpha_K$ des réels tous strictement positifs. Une distribution de Dirichlet d'exposants $\alpha_1, \alpha_2, ..., \alpha_K$ (également

appelés les hyperparamètres) possède une densité de probabilité de la forme suivante :

$$Dir(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} \theta^{\alpha_k - 1}$$

Cette densité de probabilité est définie sur le domaine du simplexe, i.e., l'ensemble des vecteurs sont définis tel que $0 \leq \theta_k \leq 1$ et $\sum_{k=1}^{K} \theta_i = 1$.

et où $B(\alpha_1, \alpha_2, ..., \alpha_K)$ est la généralisation de la fonction beta [2] de deux variables à K variables (voir annexe B.1.1) :

$$B(\boldsymbol{\alpha}) := \frac{\prod_{i=1}^{K} \Gamma\alpha_i}{\Gamma(\alpha_0)}$$

où $\alpha_0 := \sum_{i=k}^{K} \alpha_k$.

et Γ est la fonction Gamma, qui satisfait $\Gamma(x+1) = x\Gamma(x)$ et $\Gamma(1) = 1$.

Nous utiliserons dans nos travaux sur la catégorisation de textes des *a priori* modélisés par distributions de Dirichlet symétrique, avec α et β ayant une seule valeur. Les hyper-paramètres α et β spécifient la nature des *a priori* sur θ_d et ϕ_k. L'hyper paramètre α peut être interprété comme une observation *a priori* sur la taille du mélange des thèmes dans un document d [Steyvers 2007]. L'hyper paramètre β à son tour peut être interprété comme une observation *a priori* sur la taille du mélange des mots w dans un thème z [Steyvers 2007, Minka 2000].

LDA et les hyper-paramètres. Le choix de la distribution de Dirichlet est basé sur deux axes : le premier est que la distribution de Dirichlet est une distribution de probabilités sur des proportions multinomiales. C'est-à-dire au lieu de tirer directement $\mathbf{X}$ (e.g. mots) de la distribution $\boldsymbol{\alpha}$ (e.g. thème), on injecte un *a priori* avec un choix de $\alpha_1, \alpha_2, ..., \alpha_K$. L'*a priori* de Dirichlet nous dicte le choix de la distribution $\boldsymbol{\alpha}$ la plus probable de générer $\mathbf{X}$. Le deuxième axe est que le choix des hyper-paramètres α joue le rôle de régulateur pour éviter le sur-apprentissage, par exemple dans la figure 3.4 $\boldsymbol{\alpha} < 1$ nous donne une distribution uniforme des proportions. Par exemple, dans le modèle LDA, une valeur de $\boldsymbol{\alpha} > 1$, résulterait dans une taille plus grande du mélange de thèmes dans un document. Une forte valeur de $\boldsymbol{\beta}$, résulterait dans une taille plus importante du mélange de mots dans un thème. C'est-à-dire, que le thème en question pourra comporter presque tous les mots du vocabulaire V de la collection de documents.

La partie graphique de la figure 3.3 est représentée par un graphe dont les sommets (i.e. noeuds) sont des variables aléatoires $\mathcal{X} = X_1, ..., X_n$. La disposition des sommets dans la structure imposera des contraintes sur leurs dépendances (définition B.2) permettant de ce fait une factorisation de la distribution jointe en un produit de facteurs. Cette factorisation permet d'écrire la distribution jointe (e.g. équation 3.6) de façon malléable ne faisant intervenir qu'un petit nombre de variables. Une

2. L'étude de la fonction Beta est présentée dans l'annexe B.1.1

écriture plus simple (i.e courte) de la distribution jointe permet d'utiliser des algorithmes plus performants et plus rapides comparés à la palette d'algorithmes que l'on peut utiliser sur des écritures plus formelles et plus longues [Koller 2009].

Avec l'utilisation du réseau bayésien illustré dans la figure 3.3 il est possible de réécrire la distribution jointe de l'équation 3.6 comme suit :

$$p(\theta, \phi, z|w, \alpha, \beta) = \frac{p(\theta, \phi, z, w|\alpha, \beta)}{p(w, \alpha, \beta)} \tag{3.7}$$

Avec l'équation 3.7, il y a une translation de la problématique de la détection de thème vers une problématique d'inférence bayésienne où il s'agit de calculer la distribution *a posteriori* $p(\theta, \phi, z|w, \alpha, \beta)$ (se référer à l'annexe B.1.1 pour un rappel sur le raisonnement bayésien). Ceci revient à inverser le processus génératif et apprendre les quantités latentes décrites dans le modèle en tenant compte des données observées.

3.2.3 Opérationnalisation de notre approche

Le calcul de cette distribution *a posteriori* est le problème clé de LDA, car elle n'est pas calculable en temps polynomial à cause du facteur de normalisation $p(w, \alpha, \beta)$ [Heinrich 2004]. Effectuer une inférence exacte est un problème NP-difficile [Cooper 1990], qui est approximativement résolu avec l'utilisation des méthodes d'inférences approchées. Les deux méthodes les plus utilisées dans la littérature sont les méthodes d'approximation variationnelle ([Beal 2003] voir l'annexe B.1.1) et l'échantillonnage de Gibbs [Blei 2012]. Nous avons choisi d'utiliser l'échantillonnage de Gibbs et plus particulièrement une version réduite pour apprendre les distributions θ et ϕ en tirant aléatoirement un mot dans un thème et un thème dans un document selon une distribution de proabbailité jusqu'à la convergence des résultats.

3.2.3.1 Inférence avec l'échantillonnage de Gibbs

Le but de la simulation stochastique MCMC est de tirer aléatoirement des échantillons d'une distribution de probabilité $p(x)$ donnée afin d'approcher un état stationnaire de la chaîne de Markov qui correspondra à une approximation de la distribution a posteriori. Cette opération est répétée m fois sur la chaîne de Markov, avec m étant le nombre d'itérations. Dans chaque itération, l'on fait un pas dans la chaîne de Markov qui a pour état final la distribution a posteriori.

L'échantillonnage de Gibbs est l'une des méthodes de simulation stochastique de Monte-Carlo avec une chaîne de Markov (*Monte-Carlo Markov Chain*). L'échantillonnage de Gibbs est un cas particulier de l'algorithme de Métropolis [Metropolis 1953] et de son extension par Hastings [Hastings 1970]. L'appellation échantillonnage de Gibbs a été utilisée la première fois dans les travaux de Geman [Geman 1984] où il s'agissait de générer des observations à partir d'une distribution de Boltzmann.

Les avantages de l'échantillonnage de Gibbs par rapport à la méthode de Metropolis-Hastings sont : sa capacité de gestion des échantillons i.i.d. et sa capacité de simulation de distributions multidimensionnelles. Ceci fait de la méthode de Gibbs la candidate parfaite pour les modèles bayésiens hiérarchiques (définition 3.1).

Algorithme 1 Algorithme de l'échantillonnage de Gibbs

1: Initialiser aléatoirement $\boldsymbol{\theta}$
2: Initialiser $t = 0$
3: **tant que** Convergence non obtenue et $t < m$ **faire**
4:
5: $\quad \theta_1^{t+1} \sim p(\theta_1 | \theta_2^t, \theta_3^t, \ldots, \theta_d^t, \boldsymbol{X})$
6:
7: $\quad \theta_2^{t+1} \sim p(\theta_2 | \theta_1^t, \theta_3^t, \ldots, \theta_d^t, \boldsymbol{X})$
8:
9: $\quad \theta_{d-1}^{t+1} \sim p(\theta_{d-1} | \theta_1^{t+1}, \theta_2^{t+1}, \ldots, \theta_d^t, \boldsymbol{X})$
10:
11: $\quad \theta_d^{t+1} \sim p(\theta_d | \theta_1^{t+1}, \theta_2^{t+1}, \ldots, \theta_{d-1}^{t+1}, \boldsymbol{X})$
12:
13: $\theta = \theta^{k+1}$
14: $t = t + 1$

Dans l'échantillonnage de Gibbs, nous initialisons aléatoirement le vecteur $\boldsymbol{\theta}$ de paramètres et, à chaque itération, les d composantes de $\boldsymbol{\theta}$ sont générées les unes après les autres conditionnellement aux autres composantes (voir algorithme 1) et au vecteur $\boldsymbol{X}$ de données observées. Il a été prouvé théoriquement que la convergence de l'échantillonnage de Gibbs est garantie [Resnik 2010]. Cependant, il n'y a pas de méthode précise pour savoir de combien d'itérations m l'on nécessite pour atteindre un état stationnaire de la chaîne de Markov, ou de combien d'outils pour diagnostiquer la convergence. Ceci représente l'un des inconvénients majeurs de la méthode. Dans nos expérimentations nous avons choisi $m = 1000$.

3.2.3.2 Inférence avec l'échantillonnage de Gibbs réduit

Lors de l'application de l'échantillonnage de Gibbs sur le modèle LDA, nous sommes intéressés par les quantités latentes suivantes :

— la proportion θ_d des thèmes dans les documents,
— la proportion $\phi^{(z)}$ des mots dans les thèmes
— et l'étiquetage z_i de chaque mot par un thème.

Pour calculer ces quantités latentes, nous pouvons dériver les distributions conditionnelles qui leur sont associées en appliquant l'approche classique de l'échantillonnage de Gibbs telle qu'elle est décrite dans l'algorithme 1.

$$p(z_i \mid z_{-i}, \alpha, \beta, w) \tag{3.8}$$

où z_{-i} représente toutes les allocations de thèmes sauf pour z_i.

Cependant, il est à noter que les deux quantités latentes relatives aux distributions θ_d et $\phi^{(z)}$ peuvent être calculées à partir de simples statistiques que fournit

l'étiquetage thématique z_i. Nous appliquons donc l'échantillonnage de Gibbs uniquement sur l'*a posteriori* décrit dans l'équation 3.8 pour obtenir la probabilité qu'un thème z_i soit assigné à un mot w_i étant donné toutes les autres affectations de chaque thème à chaque mot dans le vocabulaire V.

Algorithme 2 Echantillonage de Gibbs appliqué à LDA

Précondition : w mots dans le corpus de document $D = (d_1, d_2, \ldots, d_n)$
Initialiser aléatoirement z
pour tout itérations **faire**
  pour tout $w \in d_i$, $d_i \in D$ et z_i dans z **faire**
    $n_{d_i,z_i} = n_{d_i,z_i} - 1$
    $n_{w,z_i} = n_{w,z_i} - 1$
    $n_{z_i} = n_{z_i} - 1$
    pour tout thème j dans K thèmes **faire**
      Calculer $P(z_i = j | z_i, w)$
    $z_i \sim p(z|.)$
    $n_{d_i,z_i} = n_{d_i,z_i} + 1$
    $n_{w,z_i} = n_{w,z_i} + 1$
    $n_{z_i} = n_{z_i} + 1$
Calculer $\phi^{(z)}$
Calculer θ_d
retour de l'algorithme : $z, \phi^{(z)}, \theta_D$

$$\hat{\phi}_j^{(w)} = \frac{n_j^{(w)} + \beta}{n_j^{(\cdot)} + V\beta} \tag{3.9}$$

$$\hat{\theta}_j^{(d)} = \frac{n_j^{(d)} + \alpha}{n_\cdot^{(d)} + K\alpha} \tag{3.10}$$

L'équation 3.9 est l'estimation bayésienne de la distribution des mots dans les thèmes.

L'équation 3.10 est l'estimation bayésienne de la distribution des thèmes dans les documents.

$$P(z_i = j \mid z_{-i}, w) \propto \frac{n_{-i,j}^{w_i} + \beta}{n_{-i,j}^{(\cdot)} + V\beta} \frac{n_{-i,j}^{d_i} + \alpha}{n_{-i,\cdot}^{d_i} + K\alpha} \tag{3.11}$$

Où $n_{-i,j}^{w_i}$ est le nombre d'occurrences du mot w_i dans le thème j. $n_{-i,j}^{(\cdot)}$ est le nombre d'occurrences de tous les autres mots dans le thème j. $n_{-i,j}^{d_i}$ est le nombre d'occurrences du thème j dans le document d_i. Enfin, $n_{-i,\cdot}^{d_i}$ est le nombre d'occurrences de tous les autres thèmes dans le document d_i. Ces notations ont été reprises des travaux de [Griffiths 2004].

La quantité dans l'équation 3.8 est alors calculée par l'équation 3.11 qui est obtenue en intégrant sur les deux distributions θ_d et $\phi^{(z)}$. Ils sont calculés à leur tour par l'équation 3.9 et 3.10. Cette approche de l'échantillonnage est appelée échantillonnage de Gibbs réduit [Griffiths 2004, Steyvers 2007], elle est décrite dans l'algorithme 2.

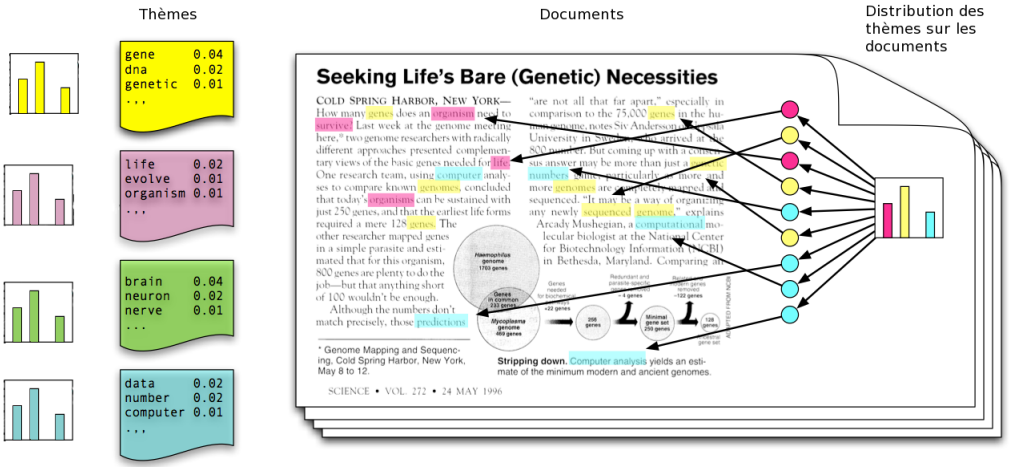


FIGURE 3.5 – Modélisation probabiliste d'un texte avec les différentes distributions des mots dans les thèmes à gauche de la figure et la modélisation de leurs proportions respective dans chaque document à droite de la figure. Illustration de [Blei 2012]

Après application de l'algorithme 2, nous obtenons la distribution ϕ des mots dans les thèmes et la distribution θ des thèmes dans les documents. L'exemple donné dans la figure 3.5 illustre la distribution de thèmes θ à droite de la figure, chaque mot est affecté à un thème z, ceci est illustré par les pastilles de différentes couleurs. La distribution des mots dans les thèmes est illustrée à gauche de la figure.

Dans cette section, nous avons abordé notre problématique d'apprentissage des thèmes à partir d'un corpus de textes. Pour cela nous avons présenté la mathématisation des corpus de textes à travers la structure de données matrice. Cette matrice soulève un problème de dimensionnalité, c'est pour cette raison que nous avons introduit les modèles génératifs au travers de l'allocation latente de Dirichlet (LDA), ce qui nous évite la manipulation de la matrice de grande dimension. LDA est un modèle graphique probabiliste représenté par un réseau bayésien qui nous permet d'appliquer un raisonnement bayésien sur la problématique d'apprentissage des thèmes. De plus, le raisonnement bayésien nous offre un cadre de travail pour utiliser les algorithmes stochastiques pour l'approximation de l'a posteriori tel que l'algorithme de Gibbs que nous avons utilisé pour calculer la distribution des thèmes dans un document. Le résultat de cette stratégie nous a permis d'obtenir une représentation thématique des textes qui est traitable et efficace. Dans la prochaine section nous allons définir un modèle de proximité en utilisant la prétopologie. Ceci afin de résoudre la deuxième problématique qui est la classification des textes en tenant compte de leur structure thématique.

3.3 La prétopologie comme modèle de proximité

Nous avons choisi d'appliquer sur une collection de documents la méthode de l'allocation latente de Dirichlet afin d'obtenir une représentation thématique où les documents sont décrits dans l'espace sémantique par des distributions de probabilités. Afin de regrouper les documents par thèmes, nous avons besoin de connaître les relations de proximités d'un document par rapport à un autre. Nous développerons cette notion de voisinage en faisant appel aux axiomes de la prétopologie.

L'observation des mouvements de la structure thématique au cours du temps au sein de l'espace sémantique est envisagée par une adaptation de la prétopologie.

> La prétopologie est un outil de modélisation mathématique du concept de proximité dans les espaces discrets. [Belmandt 2011]

Elle correspond à un affaiblissement de l'axiomatique de la topologie mathématique pour fournir un outil mathématique mieux adapté à l'analyse des phénomènes rencontrés dans des espaces discrets. Dans notre cas elle permet d'interpréter les relations thématiques entre un groupe de documents.

3.3.1 Les espaces prétopologiques comme espace sémantique

La prétopologie nous permet de concevoir des outils puissants pour la classification automatique supervisée et non supervisée grâce à des concepts comme la dilatation, l'adhérence, la fermeture, les fermés minimaux. Nous utilisons ces concepts pour structurer l'espace d'analyse des données. Dans cette section, nous présentons les définitions des concepts généraux de la théorie de la prétopologie. Le but de présenter cette série de définitions est d'introduire la liste d'outils dont nous nous servirons pour associer les textes dans un ensemble de voisinages.

Soit E un ensemble non vide dont les éléments sont des documents, et $\mathcal{P}(E)$ l'ensemble des parties de E. Nous souhaitons structurer les documents en des sous-ensembles appelés voisinages. Les documents qui sont dans le même voisinage doivent posséder la même structure thématique. Pour ce faire, nous allons utiliser les espaces prétopologiques.

Définition 3.3. On appelle espace prétopologique le triplet (E, i, a)

Où E est notre ensemble de documents, a est la fonction d'adhérence, i est la fonction d'intérieur.

Définition 3.4. Nous appelons adhérence une application $a : \mathcal{P}(E)$ dans $\mathcal{P}(E)$ tel que :

$$P_1 : a(\emptyset) = \emptyset; P_2 : \forall A \subset E, A \subset a(A) \tag{3.12}$$

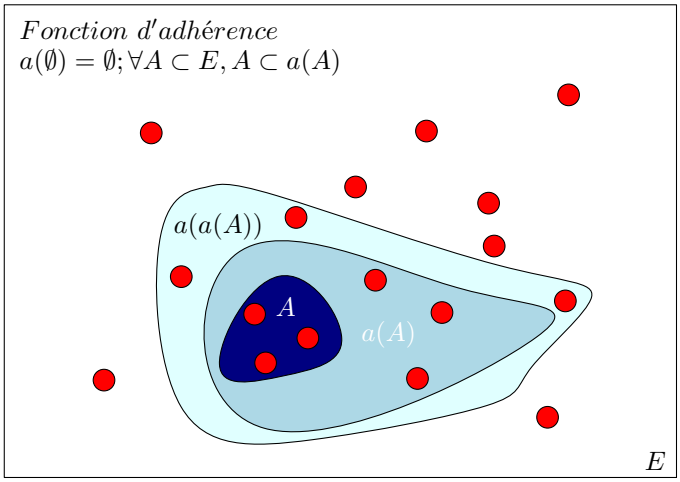


FIGURE 3.6 – Définition d'une fonction d'adhérence sur les points d'un ensemble E

L'adhérence est associée au processus de dilatation. Selon la manière avec laquelle nous l'implémentons, l'adhérence permet de récupérer pas à pas des éléments à son sous-ensemble de départ. Par exemple, dans la figure 3.6 l'ensemble de départ est en bleu foncé et l'application successive de l'adhérence nous a permis de récupérer d'autres éléments.

Définition 3.5. Nous appelons intérieur une application $i : \mathcal{P}(E)$ dans $\mathcal{P}(E)$ tel que :

$$i(E) = E; \forall A \subset E, i(A) \subset A \tag{3.13}$$

L'intérieur est associé au processus d'érosion.

L'espace prétopologique que nous avons défini à ce niveau nous ne permet pas de définir un voisinage, car les propriétés P_1, P_2 ne sont pas suffisantes. Nous devons utiliser un espace prétopologique de type $\mathcal{V}$ qui possède la propriété d'inclusion de l'adhérence d'un ensemble dans l'adhérence d'une autre. Cette dernière nous permet de définir la notion de voisinage et elle est définit comme suit :

$$P_3 : \forall A, A \subset E, \forall B, B \subset E, A \subset B \rightarrow a(A) \subset a(B) \tag{3.14}$$

En appliquant le principe de la dualité (voir équation 3.12), la propriété P_3 définie est valable aussi pour l'érosion $i(.)$. La définition d'un espace prétopologique de type $\mathcal{V}$ est donc comme suit :

Définition 3.6. (E, i, a) est dit un espace prétopologique de type $\mathcal{V}$ ssi $a(.)$ possède les propriétés P_1, P_2, P_3.

Pour les autres types d'espaces prétopologiques nous vous invitons à vous référer à [Belmandt 2011].

À partir de l'espace prétopologique de type $\mathcal{V}$ et de l'application de l'adhérence, le voisinage et base de voisinage d'un élément x, par exemple un document, est défini dans ce qui suit.

Définition 3.7. Soit (E,i,a) un espace prétopologique de type $\mathcal{V}$. $V \subset E$, est un voisinage de x ssi $\forall x \in E$, $x \in i(V)$. L'ensemble de voisinage de x est noté $\mathcal{V}(x) = \{V \subset E / x \in i(V)\}$.

À partir la définition 3.7, nous obtenons le résultat suivant dont la preuve se trouve dans [Belmandt 2011]. Soit $\forall x \in E$, un filtre des parties de E qui contiennent x. Le filtre est représenté par une fonction $\mathcal{W}(x)$. Si on définit $a(.)$ et $i(.)$ de $\mathcal{P}(E)$ dans $\mathcal{P}(E)$ alors :

$$a(A) = \{x \in E | x \in \mathcal{W}(x), V \cap A \neq \emptyset\} \quad (3.15)$$

$$i(A) = \{x \in E | x \in \mathcal{W}(x), V \subset A\} \quad (3.16)$$

Avec les définitions précédentes des applications d'adhérence et d'intérieur, $(E, i(.), a(.))$ est est un espace prétopologique de type $\mathcal{V}$ et $\forall x \in E$, les éléments de $\mathcal{W}$ sont appelés voisinage de x.

Définition 3.8. Soit $(E, i(.), a(.))$ un espace prétopologique de type $\mathcal{V}$ et $\forall x \in E$ soit $\mathcal{V}(x)$ le préfiltre des parties de E qui contiennent x. On appelle base de $\mathcal{V}(x)$ toute famille $\mathcal{B}(x)$ tel que :

$$\forall V, V \in \mathcal{V}, \exists B, B \in \mathcal{B}(x), B \subset V \quad (3.17)$$

À partir de la définition 3.8, nous obtenons le résultat suivant dont la preuve se trouve dans [Belmandt 2011]. Soit $(E, i(.), a(.))$ un espace prétopologique de type $\mathcal{V}$ et $\mathcal{B}(x)$ la base de $\mathcal{V}(x)$ alors :

$$a(A) = \{x \in E | \forall B, B \in \mathcal{B}(x), B \cap A \neq \emptyset\} \quad (3.18)$$

$$i(A) = \{x \in E | \exists B, B \in \mathcal{B}(x), B \subset A\} \quad (3.19)$$

À partir de l'ensemble des voisinages construit, nous souhaitons structurer notre espace prétopologique en un ensemble de fermés c'est-à-dire des voisinages qui n'évoluent plus en taille. Ceci est utile pour reconnaître par exemple le nombre de partitions dans une collection de documents.

3.3.2 Les fermés comme outils de caractérisation de la structure

Définition 3.9. Étant donné un espace prétopologique $(E, a(.))$, et $\forall A, A \subset E$. On nomme A comme étant un sous ensemble fermé ssi $a(A) = A$.

Le processus de dilatation généré par l'adhérence s'arrête à un instant donné et n'évolue plus. Dans ce cas, on a $a^{k+1}(A) = a^k(A)$ (voir 3.3.2). Dans ce cas l'on définit une fermeture F comme suit :

Définition 3.10. Étant donné un espace prétopologique $(E, a(.))$, et $\forall A, A \subset E$. On nomme la fermeture de A, le plus petit sous ensemble fermé de $(E, a(.))$ qui contient A. La fermeture de A est noté $F(A)$.

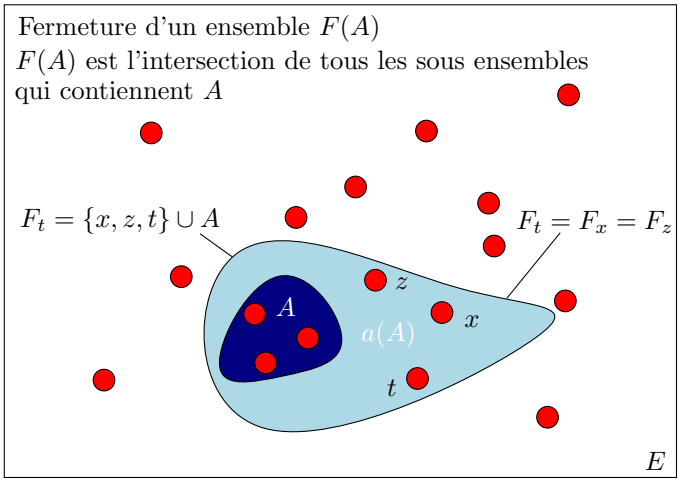


FIGURE 3.7 – Illustration du principe de la fermeture d'un ensemble

Notons que l'on nomme A comme étant un sous ensemble ouvert ssi $A = i(A)$. Dans ce travail de thèse, ne nous intéressant pas à ce type de sous ensemble. Dans l'étude du phénomène de diffusion nous nous intéresserons uniquement aux sous-ensembles qui se propagent sans diffuser vers l'extérieur. Cependant, ils peuvent toujours être influencés par d'autres diffusions. Ceci n'est pas le cas des sous-ensembles ouverts définit par $i(.)$.

Proposition 3.1. Dans tout espace prétopologique de type $\mathcal{V}$, étant donné un sous-ensemble A de E, la fermeture de A existe toujours.

Alors, dans un espace prétopologiqe de type $\mathcal{V}$, étant donné un ensemble fini E, la fermeture $F(A)$ existe toujours et peut être calculée par la propriété suivante, qui est aussi utile pour calculer la distance entre les éléments E.

$$\exists k < |E|, F(A) = a^k(A) = a(a^{k-1}(A)) \tag{3.20}$$

Dans cette sous-section, nous avons présenté notre méthode pour la représentation des textes dans un espace prétopologique. Par la suite, nous avons modélisé la notion de voisinage dans l'espace créé avec des axiomes de la prétopologie. Suite à cette modélisation, nous avons pu définir une fonction de proximités entre les documents et le nombre de partitions dans lequel ils peuvent se regrouper. Afin d'évaluer cette classification, plusieurs mesures dans l'état de l'art sont proposées.

3.3.3 Mesures d'évaluation des modèles non supervisés de classification de textes

Nous nous intéressons dans cette section aux mesures de concordance des groupes issus des résultats de classification menés par des algorithmes d'apprentissage non supervisé. L'indice de Rand ajusté et l'information mutuelle sont parmi les critères les plus utilisés pour évaluer les groupes de données. Nous présentons également la

mesure de la perplexité qui est la mesure la plus populaire pour évaluer les modèles probabilistes comme LDA.

3.3.3.1 Indice de Rand ajusté

L'indice de Rand ajusté [Hubert 1985] ou ARI pour *Adjusted Rand Index* est une adaptation ajustée de l'indice de Rand en calculant $\frac{\text{Indice}-\text{Indice attendu}}{\text{Indice maximum}-\text{Indice attendu}}$. Il est défini comme suit :

$$ARI = \frac{RI(P,Q) - E(RI(P,Q)||P|,|Q|)}{RI_{max}(P,Q) - E(RI(P,Q)||P|,|Q|)} \tag{3.21}$$

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{n_{i\circ}}{2}\sum_j \binom{n_{\circ j}}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{n_{i\circ}}{2} + \sum_j \binom{n_{\circ j}}{2}] - [\sum_i \binom{n_{i\circ}}{2}\sum_j \binom{n_{\circ j}}{2}]/\binom{n}{2}} \tag{3.22}$$

où $n_{ij}, n_{i\circ}, n_{\circ j}, n$, sont des valeurs de la table de contingence 3.1. P et Q, sont les deux groupes de données dont nous cherchons à évaluer la concordance.

$P \setminus Q$	Q_1	Q_2	$\cdots$	Q_l	Sommes
P_1	n_{11}	n_{12}	$\cdots$	n_{1l}	$n_{1\circ}$
P_2	n_{21}	n_{22}	$\cdots$	n_{2l}	$n_{2\circ}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
P_k	n_{k1}	n_{k2}	$\cdots$	n_{kl}	$n_{k\circ}$
Sommes	$n_{\circ 1}$	$n_{\circ 2}$	$\cdots$	$n_{\circ l}$	$\sum_{ij} n_{ij} = n$

TABLE 3.1 – La table de contingence. $n_{ij} = |P_i \cap Q_j|$ est le nombre de d'objets en commun entre le groupe de données P_i et Q_i.

3.3.3.2 Information mutuelle ajustée

L'information mutuelle ajustée [Vinh 2010] our AMI pour *Adjusted Mutual Information*, est une adaptation ajustée de l'information mutuelle. Comme ARI, AMI est défini selon la formule générale de l'indice ajusté qui est égale à $\frac{\text{Indice}-\text{Indice attendu}}{\text{Indice maximum}-\text{Indice attendu}}$.

$$AMI(P,Q) = \frac{MI(P,Q) - E\{MI(P,Q)\}}{\max\{H(P),H(Q)\} - E\{MI(P,Q)\}} \tag{3.23}$$

où

$$H(P) = -\sum_{i=1}^{k} \frac{n_{i\circ}}{n} \log \frac{n_{i\circ}}{n}$$

est la mesure de l'entropie ; et

$$MI(P,Q) = \sum_{i=1}^{k}\sum_{j=1}^{l} \frac{n_{ij}}{n} \log \frac{n_{ij}/n}{n_{i\circ}n_{\circ j}/n^2}.$$

MI pour *Mutual Information* représente l'information mutuelle et E représente l'espérance mathématique. P et Q sont les deux groupes de données issues de l'algorithme de classification que l'on cherche à évaluer.

La valeur de ARI et AMI est limité à 1. Plus la valeur est proche de 1, plus l'entente entre les deux partitions P et Q sur les instances qu'elles contiennent est importante.

3.3.3.3 Évaluation avec la perplexité

La mesure de la perplexité est utilisée pour évaluer la capacité d'une distribution de probabilité ou d'un modèle probabiliste à prédire la valeur d'un échantillon de données. Plus la perplexité est faible plus le modèle est meilleur. Par exemple, si la perplexité d'une implémentation LDA est faible, ceci voudrait dire que les prédictions au niveau de la distribution des mots dans les thèmes et la distribution des thèmes dans les documents est bonne.

Pour évaluer la qualité de prédiction de LDA, nous entraînons le modèle sur une collection de documents appelée $D_{\text{entrainement}}$ et nous mesurons le logarithme de la vraisemblance, étant donné le modèle entraîné, de la collection de documents mis à part, appelée D_{test}. Cette évaluation est appelée la perplexité du jeu de données de test et elle est définie comme suit :

$$\exp(-\frac{1}{N_{\text{test}}}\log p(x^{\text{test}})) \tag{3.24}$$

Dans le cas de l'évaluation de LDA, le jeu de données de test est constitué d'une collection de documents dénotés D_{test} qui n'ont pas été utilisés lors de l'entraînement de LDA.

$$p(x^{\text{test}}) = \exp(-\frac{\sum_{\tilde{m}=1}^{M}\log p(\tilde{\vec{W}}_{\tilde{m}} \mid M)}{\sum_{\tilde{m}=1}^{M} N_{\tilde{m}}}) \tag{3.25}$$

Où

$$\log p(\tilde{\vec{W}}_{\tilde{m}} \mid M) = \sum_{w=1}^{V} n_{\tilde{m}}^{w} \log \sum_{k=1}^{K} \phi_{k,w} \times \theta_{\tilde{m},k} \tag{3.26}$$

$\tilde{\vec{W}}_{\tilde{m}}$ est le vecteur de mots dans le document $\tilde{m}$. $\phi_{k,w}$ et $\theta_{\tilde{m},k}$ utilise les résultats de LDA entraîné avec la collection de documents $D_{entranement}$. $n_{\tilde{m}}^{w}$ est l'occurrence du mot w dans le document $\tilde{m}$ du jeu de données D_{test}. Enfin $N_{\tilde{m}}$ est le nombre de mots dans le document $\tilde{m}$ dans la collection D_{test}. À noter que, la littérature [Wallach 2009] offre une autre façon de calculer la perplexité avec le nombre d'échantillons s issus de l'algorithme de Gibbs. Le logarithme de la vraisemblance est décrit comme suit :

$$p(x^{\text{test}}) = \prod_{ij} \log \frac{1}{S} \sum_{s} \sum_{k} \hat{\theta}_{jk}^{s} \hat{\phi}_{x_{ij}k}^{s} \tag{3.27}$$

$$\hat{\theta}_{jk}^{s} = \frac{\alpha + n_{jk}^{s}}{K\alpha + \sum_{k} n_{jk}^{s}} \quad \hat{\phi}_{x_{ij}k}^{s} = \frac{\beta + n_{wk}^{s}}{W\beta + n_{k}^{s}}$$

Dans ce chapitre, nous avons formalisé le problème de la classification des textes par l'utilisation des modèles génératifs. Ces modèles nous font éviter la manipulation des matrices creuses à grande dimension qui représente les collections de textes. Nous avons également introduit les notions nécessaires à la modélisation d'un concept souple de proximité thématique. L'utilisation de ces notions est développée dans le chapitre suivant.

Chapitre 4

Classification des textes et gestion du passage à l'échelle

Résumé : Les collections de textes évoluent dans le temps et les thèmes qu'elles contiennent changent également : un thème majeur à un moment t peut devenir obsolète au moment $t+1$. Nous avons donc besoin d'une part d'un algorithme de classification qui intègre à la fois les thèmes majeurs et les thèmes qui sont susceptibles de l'être. D'autre part, le traitement d'une collection de textes qui évoluent dans le temps constitue un défi scientifique en raison du passage à l'échelle. Dans ce chapitre, nous proposons un algorithme de classification multicritères et une version distribuée de l'allocation latente de Dirichlet pour gérer le passage à l'échelle.

Nous adressons dans ce chapitre, le problème de la classification des textes et celui de la gestion du passage à l'échelle. Notre contribution se résume en la proposition d'une nouvelle méthode pour la classification des textes et une implémentation d'une version distribuée de l'allocation latente de Dirichlet.

La méthode de la classification de textes que nous proposons est basée sur l'analyse de la structure de l'espace sémantique avec la prétopologie (section 3.3.1). Chaque document dans cet espace est un vecteur de probabilités qui représente la distribution des thèmes présents dans le document. L'appartenance à une partition ou un groupe de documents est calculée avec deux critères : un critère sémantique et un critère quantitatif. Le critère sémantique est représenté par les thèmes majeurs dans chaque document. Le critère quantitatif est représenté par les distributions de probabilités des thèmes présents dans chaque document.

L'implémentation d'une version distribuée de l'allocation latente de Dirichlet (LDA) est basée sur la division d'un grand corpus de textes sur plusieurs unités de calculs ou plusieurs agents. Chaque agent exécute un calcul sur la portion de textes qu'il a reçu et fait remonter ses résultats à un seul superviseur. Ce dernier coordonne l'ensemble des résultats et fournit en sortie, avec la même précision qu'une version séquentielle de LDA, la distribution des thèmes dans les documents et la distribution des mots dans les thèmes.

Ce chapitre est organisé comme suit. Dans la première section, nous présentons la méthode de classification multicritères. Dans la deuxième section, nous introduisons notre proposition pour la gestion de la problématique du passage à l'échelle.

4.1 Classification multicritères

Dans cette partie, nous adressons la classification de textes avec une nouvelle méthode. Avec cette méthode nous souhaitons classifier des textes en tirant parti des thèmes majeurs (information qualitative) que partagent les textes et la distance entre deux distributions de probabilités qui sont contenues dans la représentation des textes. Cette approche est portée par notre motivation de suivre l'évolution des thèmes dans une collection de textes. Notre hypothèse est que ceci est possible si nous classifions les textes en tenant compte des thèmes majeurs à un temps t mais également des thèmes qui sont susceptibles de devenir importants à un thème t+1.

Nous décrivons notre méthode qui consiste à classifier des documents en utilisant la prétopologie, la méthode de l'allocation latente de Dirichlet (LDA) et l'algorithme K-means. Nous rappelons que l'algorithme *K-means* est une méthode de classification non supervisée qui consiste à regrouper un ensemble de données en différents sous-ensembles. Les données au sein du même sous-ensemble doivent avoir une distance minimale qui les sépare du centroïde et une distance maximale par rapport aux centroïdes des autres sous-ensembles. Cette distance est appelée mesure de similarité. Malgré qu'il soit largement utilisé pour la classification des documents, l'algorithme *K-means* possède deux limitations qui consiste en l'initialisation du nombre de sous-ensembles k avant exécution de l'algorithme, et le choix aléatoire des centroïdes des différents sous-ensembles de données.

Notre approche peut être résumé comme suit :

1. Nous calculons la distribution de thèmes pour chaque document d_i dans le corpus $\mathcal{D}$ en utilisant LDA.
2. Nous définissons deux façons d'associer les documents : une première relation binaire R_{MTP} basée sur les thèmes majeurs en commun entre deux documents, et une deuxième R_{d_H} basée sur la distance de Hellinger entre les différentes distributions de thèmes.
3. Nous définissons une fonction d'adhérence à partir des deux relations R_{MTP} et R_{d_H}.
4. Nous définissons une nouvelle distance par une application successive de l'adhérence que nous avons définie. Ceci nous permet de calculer l'ensemble des fermés et les sous-ensembles de fermés minimaux. Nous utilisons les résultats du calcul des sous-ensembles de fermés minimaux pour l'initialisation des paramètres de *K-means*.
5. Nous appliquons l'algorithme de *K-means* pour la classification des documents en utilisant d'abord les paramètres issus du calcul des sous-ensembles de fermés minimaux, ensuite la distance construite avec l'application de l'adhérence, et enfin les centroïdes qui sont aussi calculés avec l'utilisation de l'adhérence que nous avons définie.

Nous reviendrons en détails sur chaque point dans les sous-sections suivantes.

4.1.1 Structuration des documents avec LDA

Rappelons que LDA a comme donnée d'entrée une matrice de cooccurrences des mots dans les documents. En sortie, et après exécution de l'algorithme de Gibbs, le modèle fournit deux matrices : la matrice **document-thème** $\theta \in \mathbb{R}^{M \times K}$ et la matrice **thème-mot** $\phi \in \mathbb{R}^{K \times V}$.

La matrice ϕ est composée de K lignes, chaque ligne correspond au vecteur de probabilités $\phi_i \in \mathbb{R}^V$, $\forall i \in K$. Les probabilités ϕ_i, représentent les proportions de chaque mot j $\forall j \in V$ dans le thème $i \forall$ $i \in K$. Les mots avec une coordonnée j $\forall j \in V$ avec une forte valeur ϕ_{ij} sont considérés comme des mots représentatifs du thème i. En parcourant ces mots, nous pouvons inférer la représentation sémantique de chaque thème sans regarder le contenu de chaque document.

La matrice $\theta \in \mathbb{R}^{M \times K}$ est composée de M lignes, chaque ligne correspond au vecteur de probabilités $\theta_i \in \mathbb{R}^K, \forall i \in K$. Les probabilités dans θ_i représentent les proportions de chaque thème dans le document m $\forall m \in M$. Les documents avec une forte valeur θ_{im} peuvent être représentés par le thème m. Les documents avec une faible valeur θ_{im} quelque soit le thème i peuvent être considérés comme un "bruit" dans la collection de documents D. Nous pouvons inférer l'appartenance de chaque thème à chaque document en parcourant les distributions de probabilités $\theta_i \in \mathbb{R}^K$.

En utilisant LDA, nous pouvons donc caractériser un document selon sa distribution de thèmes mais nous pouvons également l'étiquetter avec un thème possédant une forte valeur de probabilité.

4.1.2 Association des documents par des relations binaires

Dans cette sous-section, nous définissons deux façons d'associer les documents. La première est basée sur les thèmes possédant une forte probabilité que deux documents partagent et la deuxième est basée sur la distance entre leur distribution de thèmes respective.

Association des documents par les thèmes majeurs

D'abord, nous associons deux documents d_n, d_m s'ils partagent des thèmes possédant une probabilité supérieur à un seuil p_0. Ce seuil est choisi au moment des expériementations et il est fixé après plusieurs itérations. Nous appelons ces thèmes des thèmes majeurs et nous les définissons comme suit :

Définition 4.1 (Ensemble des thèmes majeurs)**.** Dans la collection de documents D, nous nous référons à l'ensemble des thèmes majeurs dénoté $MTP(d_i)$, comme l'ensemble des thèmes ayant une probabilité supérieure au seuil p_0 parmi la distribution de thèmes dans le document d_i. Où $p_0 \leq 1/K$, et K est le nombre de thèmes. $MTP(d_i) = \{k \mid \theta_{ik} = \max_j \ \theta_{ij} \text{ et } \theta_{ik} \leq p_0\}$.

L'association des documents par les thèmes majeurs est définie comme suit

Définition 4.2. Un document d_m est associé à un document d_n par la relation binaire R_{MTP} si $MTP(d_m) \cap MTP(d_n) = \emptyset$

Association des documents par la distance de Hellinger

Ensuite, nous associons deux documents d_n, d_m si tous les thèmes qu'ils contiennent sont proches ou similaires. Nous pouvons utiliser la distance entre deux vecteurs de probabilités qui représente à chacun la distribution des thèmes dans un document. En considérant la distribution de probabilités d'un document comme un vecteur, nous avons le choix entre plusieurs distances, par exemple, la distance euclidienne, le coefficient de Jaccard, le coefficient de la corrélation de Pearson, etc. Néanmoins, et puisque nous travaillons avec des distributions de probabilités il est préférable de choisir des distances qui sont plus adaptées, comme la divergence de Kullback-Leibler, la distance de Bhattacharyya, la distance de Hellinger, etc. Nous avons choisi de travailler avec la distance de Hellinger, car c'est métrique qui est utilisée pour mesurer la déviation entre deux distributions de probabilité, elle est facile à calculer et elle est surtout définie dans $[0, 1]$.

Définition 4.3. Étant donné deux distributions de probabilités $P = (p_1, \ldots, p_k)$ et $Q = (q_1 \ldots q_k)$, la distance de Hellinger entre ces dernières est définie de la manière suivante :

$$d_H(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} (\sqrt{p_i} - \sqrt{q_i})^2} \tag{4.1}$$

La distance de Hellinger est liée à la norme euclidienne de la différence entre les racines carrées des vecteurs en considération, i.e.

$$d_H(P, Q) = \frac{1}{\sqrt{2}} \left\| \sqrt{P} - \sqrt{Q} \right\|_2.$$

La distance de Hellinger satisfait l'inégalité suivante $0 \leq d_H(P, Q) \leq 1$. Cette distance est une métrique pour mesurer la déviation de la distribution P de la distribution Q. La distance est égale à 0 si $P = Q$. La distance est égale à 1 si P est complètement disjointe de Q. Plus la valeur de la distance de Hellinger est faible, moins la première distribution dévie de la deuxième.

Nous proposons d'utiliser la distance d_H pour mesurer la similarité entre deux documents d_m, d_n représentés respectivement par leurs distributions de probabilités $\theta_{d_m}, \theta_{d_n}$. À partir de cette distance, l'association des documents est définie comme suit :

Définition 4.4. Un document d_n est associé à un docuemnt document d_n par la relation binaire R_{d_H} si $d_H(d_m, d_n) \leq d_0$, tel que $0 \leq d_0 \leq 1$, d_0 est le seuil choisi après plusieurs itérations dans nos expérimentations.

4.1.3 Construction de la fonction d'adhérence

À partir des relations binaires R_{MTP} et R_{d_H}, nous pouvons construire une base de voisinage (définition 3.8) d'un ensemble de documents et à partir de cette dernière construire une fonction d'adhérence. Nous proposons pour la construction de la base

Algorithme 3 Création d'une base de voisinage avec LDA

Précondition : Distribution θ des thèmes dans les documents.
Précondition : Corpus de documents D.
Précondition : R_{MTP}, R_{d_H} : famille de relations binaires.
pour tout relation $R_i \in \{R_{MTP}, R_{d_H}\}$ **faire**
  pour tout document $d_l \in D$ **faire**
    pour tout document $d_j \in D$ **faire**
      si $R_i(d_l, d_j)$ **alors**
        ajouter d_j à $\mathcal{B}_i[d_l]$
retour $\mathcal{B} = [\mathcal{B}_1, \mathcal{B}_2]$

de voisinage $\mathcal{B}$ l'algorithme 3 où l'on parcourt la collection de documents D et on rajoute un document à $\mathcal{B}$ s'il est mis en relation avec d'autres documents par les relations binaires R_{MTP} et R_{d_H}.

Algorithme 4 Construction de la fonction d'adhérence en utilisant LDA

Précondition : Base de voisinage $\mathcal{B} = [\mathcal{B}_1, \mathcal{B}_2]$.
Précondition : Corpus de documents D.
$aA() = A$
pour tout document $d_l \in D$ **faire**
  si $A \cap \mathcal{B}_1[d_l] \neq \emptyset$ OU $A \cap \mathcal{B}_2[d_l] \neq \emptyset$ **alors**
    ajouter d_l au sous-ensemble aA
retour l'adhérence aA().

La construction de la fonction d'adhérence est explicité dans l'algorithme 4. Deux adhérences y sont proposées : une adhérence dans une prétopologie forte (avec une opération d'intersection) et une adhérence dans une prétopologie faible (avec une opération d'union).

Etant donnée R_{MTP} et R_{d_H} qui nous avons définir sur l'ensemble fini E de documents. Et considérant $\forall i = 1, 2, \ldots, n, \forall x \in E$, le voisinage de document $V_i(x)$ est définit par :

$$V_i(x) = \{y \in E | x \ R_{MTP} \ y\} V_i(x) = \{y \in E | x \ R_{d_H} \ y\} \tag{4.2}$$

Nous appellons alors une prétopologie forte, induite par la famille R_{MTP} et R_{d_H}, la prétopologie définie sur E par $a_s(.)$ utilisant l'opérateur d'intersection. [1]

$$a_s(A) = \{x \in E | \forall i = 1, 2, \ldots, n, V_i(x) \cap A \neq \emptyset\} \tag{4.3}$$

De la même façon, nous appelons une prétopolgie faible, induite par la famille R_{MTP} et R_{d_H} , la prétopologie définie sur E par $a_w(.)$ utilisant l'opérateur d'union.

$$a_w(A) = \{x \in E | \exists i = 1, 2, \ldots, n, V_i(x) \cap A \neq \emptyset\} \tag{4.4}$$

Proposition 4.1. $a_s(.)$, $a_w(.)$ établissent une structure prétopologique sur E et les espaces prétopologique $(E, a_s(.))$, $(E, a_w(.))$ sont de type $\mathcal{V}$.

1. la lettre s dans $a_s(.)$ est pour dire *strong* en anglais (forte). La lettre w dans $a_w(.)$ est pour dire *weak* en anglais (faible).

4.1.4 Construction d'une distance avec l'application de l'adhérence

L'utilisation de *K-means* requière le calcul des centroïdes de chaque partition de document. Les coordonnées du centroïde dans un espace multidimensionnel sont calculées par la moyenne arithmétique de chaque dimension sur tous les points d'une partition. Ce calcul n'est pas effectif dans l'analyse thématique catégorielle. La fonction d'adhérence que nous avons définie est utilisée pour examiner la similarité entre deux documents, et ce en utilisant des données catégorielles (thèmes majeurs), mais aussi des données numériques (la distance entre deux distributions de probabilités). La fonction d'adhérence peut donc contribuer à améliorer les résultats de classification de *K-means*.

Définition 4.5 (Distance construite à partir de l'adhérence)**.** Pour un ensemble fini E, nous définissons la distance, établie à l'aide de l'application de la fonction d'adhérence sur deux sous-ensembles $A, B \subset E$, comme suit :

$$k_0 = \min(\min\{k \mid A \subset a^k(B)\}, \infty) \tag{4.5}$$

$$k_1 = \min(\min\{k \mid B \subset a^k(A)\}, \infty) \tag{4.6}$$

$$\delta(A, B) = \min(k_0, k_1) \tag{4.7}$$

où $a^k(\cdot) = a^{k-1}(a(\cdot))$.

k est le nombre de pas dans le processus de dilatation décrit par l'adhérence. $A \subset a^k(B)$ est le nombre de pas qu'il a fallu pour que A soit contenu dans l'adhérence de B après un nombre k de pas.

Définition 4.6 (Distance intra-points)**.** Nous appelons $D_A(x)$ la distance entre un point x et les autres points dans un ensemble A

$$D_A(x) = \frac{1}{|A|} \sum_{y \in A} \delta(x, y). \tag{4.8}$$

Dans le cas où les deux sous-ensembles A et B sont réduits à un seul élément x et y, la distance est calculée avec $\delta(x, y)$. Pour la classification des documents avec *K-means*, nous utilisons la distance $\delta(x, y)$ pour calculer la distance entre deux documents (chaque document est représenté par sa distribution de thèmes est un point $x \in E$). Nous utilisons la distance intra-points $D_A(x)$ pour claculer le centroïde de A. Par exemple, x_0 est choisi comme centroïde de A si $D_A(x_0) = \min_{x \in A} D_A(x)$.

4.1.4.1 Analyse de la structure de l'espace des documents avec les fermés minimaux

Les deux contraintes de *k-means* sont le nombre de partitions qui doit être prédéterminé et le choix aléatoire du centroïde initial pour le calcul des partitions. La prétopologie nous permet de contourner ces contraintes en utilisant les résultats a posteriori du calcul des sous-ensembles de fermés minimaux.

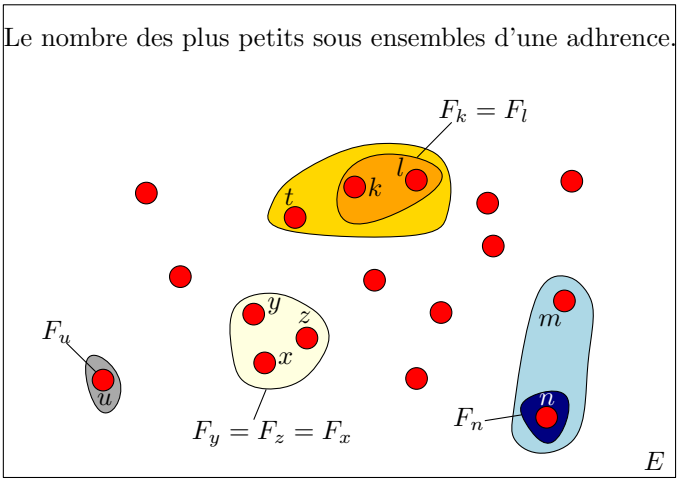


FIGURE 4.1 – Illustration de l'application de la fonction d'un fermé élémentaire F_x pour obtenir les fermés minimaux $\mathcal{F}_m = F_{mj}, j = 1, \ldots, k$

Définition 4.7. On appelle fermé élémentaire, noté F_x la fermeture d'un singleton $\{x\}$ de E. On note $\mathcal{F}_e(E, a)$ ou $\mathcal{F}_e$, l'ensemble des fermés élémentaire de E. Alors sur un espace prétopologique de type $\mathcal{V}$, nous obtenons :

$$\forall x \in E, \exists F_x : \text{fermeture de}\{x\} \tag{4.9}$$

$$\mathcal{F}_e(E, a) = \{F_x | x \in E\} \tag{4.10}$$

Définition 4.8. On appelle fermé minimal de E, tout élément de $F(E, a)$, minimal au sens de l'inclusion. C'est-à-dire qu'un fermé minimal ne peut pas contenir un fermé plus petit que lui. L'ensemble des fermés minimaux est noté : $\mathcal{F}_m(E, a)$ ou $\mathcal{F}_m$.

Nous dénotons par $\mathcal{F}_m = \{F_{m_j}, j = 1, 2, \ldots, k\}$, la famille des fermés minimaux l'ensemble des sous-ensembles minimaux fermés dans $\mathcal{F}_e$. Un résultat important est que tout fermé minimal est obligatoirement élément de $\mathcal{F}_e$, c'est-à-dire un fermé élémentaire. Déterminer les fermés minimaux revient donc à explorer les éléments de $\mathcal{F}_e$ et en extraire les éléments minimaux par la relation d'inclusion. La figure 4.1 illustre ce processus.

En exécutant l'algorithme 5 inspirée des travaux de [Levorato 2008] nous obtenons la famille des sous-ensembles de fermés minimaux. Cette famille caractérise la structure sous-jacente de l'ensemble E des données. Le nombre des sous-ensemble de fermés minimaux nous donne le nombre de partitions pour initialiser *k-means*. Quand au centroïde initial il peut être calculer par l'utilisation de la distance intra-points et ce pour chaque sous-ensemble minimal fermé telque c'est défini dans 4.9.

Définition 4.9. Pour tout fermé minimal F_{m_j} tel que $F_{m_j} \in \mathcal{F}_m$, x_0 est le centroïde de F_{m_j} Si $D_{F_{m_j}}(x_0) = min_{x \in F_{m_j}} D_{F_{m_j}}(x)$.

Algorithme 5 Algorithme de construction des fermés minimaux.

Précondition : Adhérence aA().
Précondition : Corpus de documents D.
Calculer la famille des fermés élémentaires $\mathcal{F}_e$
$\mathcal{F}_m = \emptyset$
tant que $\mathcal{F}_e \neq \emptyset$ **faire**
    Choisir F tel que $F \subset \mathcal{F}_e$
    $\mathcal{F}_e = \mathcal{F}_e - F$
    Minimum = VRAI
    $\mathcal{F} = \mathcal{F}_e$
    tant que $\mathcal{F} \neq \emptyset$ et Minimum est VRAI **faire**
        Choisir G tel que $G \in \mathcal{F}$
        si $G \subset F$ **alors**
            Minimum = FALSE
        sinon
            si $F \subset G$ **alors**
                $\mathcal{F}_e = \mathcal{F}_e - G$
        $\mathcal{F} = \mathcal{F} - G$
si Minimum est VRAI et $F \notin \mathcal{F}_m$ **alors**
    $\mathcal{F}_m = \mathcal{F}_m \cap F$
retour $\mathcal{F}_m$. Sous ensembles de fermés minimaux.

Algorithme 6 Algorithme de la classification mutli-critères

Précondition : Corpus de documents D.
Calculer θ_D avec l'échantillonnage de Gibbs.
Création de la base de voisinage $\mathcal{B}$ à partir de θ_D.
Calculer l'adhérence $aA()$ sur la base $\mathcal{B}$.
Calculer les fermés minimaux $\mathcal{F}_m$ avec l'adhérence $aA()$.
$k = |\mathcal{F}_m|$: nombre de regroupements
pour tout les regroupements k où $i < k$ **faire**
    pour tout $F_{m_i} \in \mathcal{F}_m$ **faire**
        Ajouter à M le centroïde de F_{m_i}
tant que il y a changement dans les centroïdes **faire**
    pour tout $x \in E - M$ **faire**
        Calculer la distance par adhérence $\delta(x, m_i) \forall i = 1, \ldots, k$
        Trouver le centroïde m_0 le plus représentatif des données tel que
        $\delta(x, m_0) = \min \delta(x, m_i) \forall i = 1, \ldots, k$
        $F_{m_0} = F_{m_0} \cup \{x\}$
    Recalculer les centroïdes
retour Les regroupements de données $\{F_1, F_2, \ldots, F_k\}$

4.1.4.2 L'algorithme de la classification multi-critère

Notre proposition pour une classification multi-critère d'un document est décrite dans l'algorithme 6. Premièrement, LDA est appliquer sur une collection de documents pour calculer la distribution de thèmes dans chaque document représenté par la distribution de probabilité θ. Deuxièmement, les thèmes majeurs de la collection ainsi que la distance de Hellinger entre les distributions de probabilités sont utilisés pour associer les documents par des relations binaires. Ces relations binaires sont utilisées pour construire un espace prétopologique de type $\mathcal{V}$ que nous avons utilisé pour avoir des résultats préliminaires de classification et pour déterminer le nombre de partitions. Troisièmement, nous avons utiliser l'algorithme *K-means* pour classifier les documents en utilisons deux distances construites à partir de l'application successive de l'adhérence. La première distance est utilisée pour rapprocher les do-

cuments et la deuxième est utilisée pour calculer les centroïdes de chaque partition de documents.

4.1.4.3 Illustration avec un corpus Twitter

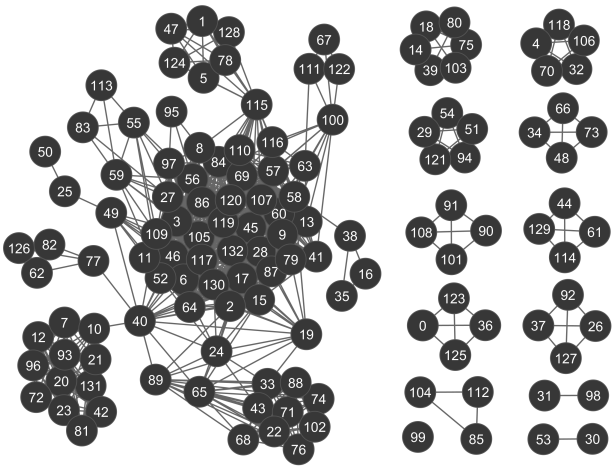


FIGURE 4.2 – Illustration du réseau des 133 utilisateurs avec deux types de relations qui les lient, la première est basée sur la distance de Hellinger ($distance \leq 0.15$) et les thèmes les plus discutés (avec une probabilité ≥ 0.15)

Twitter est un réseau social de micro blogging, qui permet de partager des messages courts, de 140 caractères maximum, appelés *tweets*. Ces *tweets* sont diffusés à un large public. Pour évaluer notre approche multicritères pour la classification des documents, nous avons choisi de l'appliquer sur la problématique de regroupement des utilisateurs selon leurs différents centres d'intérêt. Chaque utilisateur est considéré comme un document où l'on rassemble tous ses *tweets*. Nous avons choisi de glaner les *tweets* de 133 utilisateurs qui correspondent donc à 133 documents.

Les 133 utilisateurs ont été choisis selon deux critères : l'hétérogénéité du contenu qu'ils partagent et leur popularité au sein du réseau social *Twitter*. Nous avons utilisé un logiciel sous forme de service (API), fourni par le site Twitter, pour collecter les messages entre le début du mois de janvier et la fin du mois de février. Nous avons collecté dans les $D = 133$ documents, un nombre $N = 158,578$ de mots avec une moyenne de 1,192 mots par document et une taille du vocabulaire $V = 29,104$ mots uniques. Un prétraitement des données collectées a été nécessaire pour enlever les mots vides en information (*stop-words* en anglais), les liens web.

Nous avons, par la suite, appliqué LDA en utilisant l'algorithme de Gibbs réduit (voir l'algorithme 2) pour obtenir la distribution θ des thèmes dans les documents et la distribution ϕ des mots dans les thèmes. De la liste des 20 thèmes que nous avons calculés avec notre implémentation de LDA, nous avons présenté dans la table 4.1 deux thèmes. Le premier correspond aux attaques terroristes contre le journal satirique Charlie Hebdo qui se sont déroulées à Paris le 7 janvier 2015. Le second

TABLE 4.1 – Distribution des mots dans les thèmes ϕ et les utilisateurs qui relatent les thèmes en question

Thème 3				
Words	**Prob.**	**Utilisateurs**	**ID**	**Prob.**
paris	0.008	GStephanopoulos	42	0.697
charliehebdo	0.006	camanpour	23	0.694
interview	0.006	AriMelber	12	0.504
charlie	0.005	andersoncooper	7	0.457
attack	0.005	brianstelter	20	0.397
warisover	0.004	yokoono	131	0.362
french	0.004	piersmorgan	96	0.348
today	0.004	maddow	72	0.314
news	0.004	BuzzFeedBen	21	0.249
police	0.003	MichaelSteele	81	0.244

Thème 10				
Words	**Prob.**	**Utilisateurs**	**ID**	**Prob.**
ces	0.010	bxchen	22	0.505
people	0.007	randizuckerberg	102	0.477
news	0.006	NextTechBlog	88	0.402
media	0.006	lheron	71	0.355
tech	0.006	LanceUlanoff	68	0.339
apple	0.006	MarcusWohlsen	74	0.339
facebook	0.005	marissamayer	76	0.334
yahoo	0.005	harrymccracken	43	0.264
app	0.005	dens	33	0.209
google	0.004	nickbilton	89	0.204

thème correspond à la conférence CES (*Consumer Electronic Show*), qui se déroule chaque année, où l'on expose les nouvelles technologies destinées au grand public pour les années à venir.

Utilisateur ID 02	
Thème	**Prob.**
10	0.090
16	0.072
12	0.065
18	0.064
smel 0	0.058

Utilisateur ID 12	
Thème	**Prob.**
3	0.504
19	0.039
10	0.036
15	0.035
13	0.032

Utilisateur ID 22	
Thème	**Prob.**
10	0.506
3	0.036
19	0.034
14	0.031
4	0.03

Utilisateur ID 53	
Thème	**Prob.**
17	0.733
1	0.017
18	0.016
13	0.016
11	0.015

Utilisateur ID 75	
Thème	**Prob.**
19	0.526
2	0.029
3	0.029
5	0.028
105	0.028

Utilisateur ID 83	
Thème	**Prob.**
8	0.249
0	0.084
11	0.06
7	0.045
12	0.043

TABLE 4.2 – Distribution des thèmes par utilisateur

Chaque thème est représenté avec une distribution de mots. Pour chaque thème, nous avons une liste d'utilisateurs qui lui est associée. Chaque utilisateur est identifié avec un ID, qui a une valeur qui va de 0 à 132, et une liste de thèmes qui lui est associée avec un ordre de probabilités (voir la table 4.2). La liste des thèmes qui lui sont associés est considérée comme ses centres d'intérêt. Les thèmes 3 et 10 sont d'un intérêt particulier vu qu'un nombre important des 133 utilisateurs ont tweeté à propos de ces thèmes. Ceci s'explique par le fait que les deux événements se sont

déroulés au moment de la collecte des *tweets*.

Dans la table 4.2, nous avons présenté un exemple de 8 utilisateurs avec la liste des thèmes représentés par la probabilité issue de la distribution θ. Une forte probabilité $\theta_{i,j}$ indique que l'utilisateur ou document i est fortement relié au thème j. Par exemple, l'utilisateur 12 est fortement relié au thème 3, l'utilisateur 22 est fortement relié au thème 10, et ainsi de suite. Notons que les utilisateurs avec des faibles valeurs de probabilités $\theta_{i,j}$ sont qualifiés comme un bruit et ne sont pas reliés à aucun thème. Observons qu'avec cette analyse préliminaire des résultats, nous avons pu établir un premier regroupement des utilisateurs en considérant la valeur de $\theta_{i,j}$.

Dans la table 4.3, nous avons regroupé les utilisateurs selon les thèmes majeurs qui leur sont reliés. Ces thèmes majeurs constituent leurs centres d'intérêt les plus pertinents. Nous avons identifié les groupes d'utilisateurs selon deux intervalles, le premier avec une valeur de $\theta_{i,j} \leq 0.3$ et le deuxième avec une valeur de $0.15 \leq \theta_{i,j} \leq 0.3$. Ce regroupement a été produit de façon manuelle. Nous présentons, dans ce qui suit, le résultat de notre approche de classification qui prend en compte les thèmes majeurs, mais aussi la distance entre la distribution de tous les thèmes affectés à chaque utilisateur.

Thèmes majeurs	**prob $\geq$ 0.3**	**0.15 $<$ prob $<$ 0.3**
Thème 0	112,85,104	-
Thème 1	44,129,114	61
Thème 2	101,108,91	90
Thème 3	42,23,12,7,20, 131,96,72	21,81,93,10
Thème 4	125,36,123,0	-
Thème 5	82,126	62
Thème 6	127,37,26	92
Thème 7	118,106,32	70,4
Thème 8	113	83,55,59
Thème 9	67,122	111,100
Thème 10	22,102,88,71,74, 68,76	43,89,33,65
Thème 11	54,51,121	29,94
Thème 12	50	12
Thème 13	16,35	38
Thème 14	31,98	-
Thème 15	66,73,34,	48
Thème 16	99	-
Thème 17	53,30	-
Thème 18	47,128,1,124,5	78,115
Thème 19	14,80,39,75,18,103	-
$\emptyset$	**utilisateurs restants (probabilité $<$ 0.15)**	

TABLE 4.3 – Classification des documents basée sur leurs thèmes majeurs

Après la dernière classification manuelle, nous avons procédé à l'exécution de l'algorithme *K-means* en utilisant la distance de Hellinger. Par le choix de la distance de Hellinger, comme mesure de similarité dans l'algorithme *K-means*, nous voulions avoir une valeur numérique qui décrit la proximité entre deux utilisateurs en se basant sur la distribution de probabilité des thèmes dont ils parlent ou *tweetent*.

La table 4.4 présente les groupes calculés par *K-means* avec $k = 13$ et un choix

Partition	Utilisateurs	Thèmes majeurs
1	67, 111, 122	TP 9 (0.423)
2	34, 48, 66, 73	TP 15 (0.315)
3	10, 22, 33, 43, 65, 68, 71, 74, 76, 88, 89, 98, 102	TP 10 (0.305)
4	26, 92	TP 6 (0.268)
5	16, 35, 44, 90, 91, 101, 108, 114, 129	TP 2 (0.238)
6	4, 32, 70, 106, 118	TP 7 (0.345)
7	37, 127	TP 6 (0.580)
8	14, 18, 39, 75, 80, 103	TP 19 (0.531)
9	1, 5, 47, 78, 124, 128	TP 18 (0.453)
10	30, 53	TP 17 (0.711)
11	7, 12, 20, 21, 23, 42, 72, 81, 93, 96, 131	TP 3 (0.409)
12	0, 31, 36, 82, 123, 125	TP 4 (0.310)
13	Utilisateurs restants	$\emptyset$

TABLE 4.4 – Sortie de l'algorithme de *K-means* en utilisant la distance de Hellinger

de centroïdes aléatoires. Notons que pour chaque groupe, nous avons calculé les thèmes majeurs en nous basant sur la valeur moyenne de $\theta_{i,j}$ de chaque groupe.

Les différentes exécutions de l'algorithme *K-means* avec différentes configurations (i.e. nombre k de groupes ainsi que des centroïdes aléatoires) nous ont donné différentes partitions des utilisateurs. De ce fait, le choix des centroïdes ainsi que le choix du nombre de groupes sont importants pour la précision des résultats que l'on peut obtenir.

Pour résoudre ce problème, nous utiliserons les outils d'extractions des caractéristiques, développés dans 4.1.4.1, pour initialiser les paramètres de *K-means*. Pour rappel, le nombre des sous-ensembles des fermés minimaux nous donnera le nombre k de groupes et la distance intérieure calculée avec l'adhérence nous donnera le centroïde de chaque groupe.

	***K-means* & Hellinger**		**Algorithme MCPTM**	
Partition	**Utilisateurs**	**Thème**	**Utilisateurs**	**Thème**
1	0,36,123,125	TP 4 (0.457)	0,36,123,125	TP 4
2	4,32,70,10,118	TP 7 (0.345)	4,32,70,10,118	TP 7
3	14,18,39,75,80,103	TP 19 (0.531)	14,18,39,75,80,103	TP 19
4	26,37,92,127	TP 6 (0.424)	26,37,92,127	TP 6
5	29,51,54,94,121	TP 11 (0.345)	29,51,54,94,121	TP 11
6	30,53	TP 17 (0.711)	30,53	TP 17
7	31	TP 14 (0.726)	31,98	TP 14
8	34,48,66,73	TP 15 (0.315)	34,48,66,73	TP 15
9	44,61,114,129	TP 1 (0.413)	44,61,114,129	TP 1
10	85,104,112	TP 0 (0.436)	85,104,112	TP 0
11	67,90,91,101,108	TP 2 (0.407)	90,91,101,108	TP 2
12	99	TP 16 (0.647)	99	TP 16
13	Utilisateurs restants	$\emptyset$	Utilisateurs restants	$\emptyset$

TABLE 4.5 – Result from *K-means* algorithm using Hellinger distance and MCPTM

D'abord, pour obtenir la liste des sous-ensembles de fermés minimaux, nous devons construire un espace prétopologique avec les résultats de l'analyse sémantique que nous avons effectuée sur les documents (pour un rappel sur la construction d'un

espace prétopologique à partir des résultats de l'analyse sémantique latente, voir la section 3.3).

Ensuite, nous devons concevoir une mesure de similarité entre deux documents (utilisateurs) dans l'espace prétopologique. La distance est construite avec une adhérence faible qui est à son tour établie avec deux relations binaires.

En préambule, nous devons donc définir deux relations binaires : la première est la relation R_{MTP} basée sur les thèmes majeurs. Cette relation nous a permis d'obtenir la classification présentée dans la table 4.3. La deuxième est la relation R_{d_H}, basée sur la distance de Hellinger et qui nous a permis d'obtenir la classification présentée dans la table 4.4. Une fois ces étapes satisfaites, nous pouvons exécuter l'algorithme 5 pour obtenir la liste des sous-ensembles de fermés minimaux et utiliser l'adhérence pour obtenir les centroïdes tels que nous l'avons décrit dans la définition 4.9.

Enfin, nous pouvons exécuter l'algorithme 6 pour la classification multicritères, appelée MCPTM. Les résultats de cette exécution sont présentés dans la table 4.5. Nous observons une forte similarité entre les résultats des deux méthodes quand on utilise un nombre de groupes k et un centroïde initial, tous deux tirées de l'analyse de l'espace sémantique avec la prétopologie. Nous concluons à cette étape de l'analyse que la distance construite avec la fonction d'adhérence est aussi efficace que la distance de Hellinger.

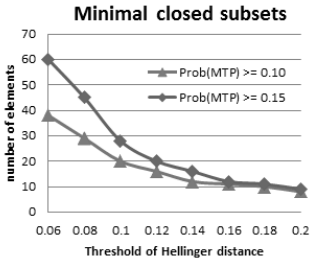


FIGURE 4.3 – Le nombre d'éléments dans les sous-ensembles de fermés minimaux avec différents seuils : p_0 pour R_{MTP} et d_0 pour R_{d_H}

La figure 4.3 montre comment nous avons sélectionné les caractéristiques extraites pour *K-means*. Ceci, en faisant varier deux seuils : le seuil p_0 pour R_{MTP} et le seuil d_0 pour R_{d_H}. Plus les seuils ont une grande valeur, plus la sélection des caractéristiques est agressive.

Pour mener nos expérimentations, nous avons choisi un $p_0 = 0.15$ et un $d_0 = 0.15$. Un utilisateur (document) i est mis en relation avec un utilisateur (document) j s'ils ont le même thème majeur avec une probabilité ≤ 0.15 ou une distance de Hellinger $d_H(\theta_i, \theta_j \leq 0.15)$.

Dans la figure 4.2, nous avons illustré le regroupement des 133 utilisateurs en 13 composantes calculées avec une adhérence faible, chaque composante du réseau de la figure 4.2 représente un sous-ensemble de fermé minimaux. Ces 13 groupes représentent le nombre k. Nous avons utilisé la distance intérieure construite avec

l'adhérence pour calculer les centroïdes initiaux pour *K-means* et nous avons obtenu les résultats suivants :

$$\{0, 52, 4, 14, 26, 29, 30, 31, 34, 44, 85, 90, 99\}$$

Notons que le composant numéro 13 du réseau est le plus large, il contient 89 noeuds avec plusieurs liens faibles, et quelques composantes fortement connexes. Nous voulions extraire automatiquement ses composantes fortement connexes, et avons donc appliqué une autre itération de *K-means* sur le groupe numéro 13. Après une analyse prétopologique nous avons obtenu un $k = 9$, et calculé les centroïdes suivants :

$$\{23, 82, 113, 67, 22, 50, 16, 47, 2\}$$

Partition	**Utilisateurs**	**Thème majeur**
13.1	7, 12, 20, 21, 23, 42, 72, 81, 93, 96, 131	TP 3 (0.409)
13.2	62, 77, 82, 126	TP 5 (0.339)
13.3	27, 55, 59, 83, 113	TP 8 (0.218)
13.4	67, 111, 122	TP 9 (0.422)
13.5	22, 33, 43, 65, 68, 71, 74, 76, 88, 89, 102	TP 10 (0.330)
13.6	50	TP 12 (0.499)
13.7	16, 35	TP 13 (0.576)
13.8	1, 5, 47, 78, 124, 128	TP 18 (0.453)
13.9	Utilisateurs restants	$\emptyset$

TABLE 4.6 – Les résultats de *K-means* utilisant la distance de Hellinger sur le groupe numéro 13 (89 utilisateurs).

Les résultats de la deuxième itération de *K-means* sont présentés dans la table 4.6. L'idée d'associer la prétopologie avec *K-means* a été proposé par [Thanh Van Le 2007], les auteurs ont aussi proposé des pistes pour construire une distance à partir de la fonction d'adhérence pour les données quantitatives et numériques. Notre approche est différente sur deux points : d'abord, nous avons exploité l'analyse sémantique latente pour construire un espace prétopologique à partir des informations structurelles données par LDA. Ensuite, nous avons démontré que la prétopologie peut être utilisée pour une classification multicritère en définissant une mesure de la similarité basée sur une fonction d'adhérence construite sur plusieurs relations.

Dans cette partie, nous avons classifié des documents selon deux critères : le premier est basé sur les thèmes majeurs dans un document (critère qualitatif). Le deuxième est basé sur la distance de Hellinger (critère quantitatif). Nous avons appliqué notre approche sur des données de Twitter, où nous avons considéré tous les messages d'un utilisateur comme un document. Ceci dans le but de regrouper les utilisateurs selon leur centre d'intérêt.

Avec notre méthode nous avons pu construire un réseau avec plusieurs relations (*multi relation networks*) en choisissant un ensemble de relations et en appliquant

une prétopologie forte et faible. Cependant, l'analyse que nous avons menée jusqu'ici n'est pas suffisante pour témoigner de l'efficacité de notre algorithme MCPTM. Nous présentons donc, dans la partie qui suit, une évaluation en comparant les différentes mesures de similarité ainsi que les initialisations des paramètres de *K-means*.

4.1.4.4 Évaluation

Nous avons mené l'évaluation de notre algorithme MCPTM comme suit : d'abord, nous avons examiné trois mesures de similarités différentes à savoir : la distance euclidienne, la distance de Hellinger et la distance construite avec la fonction d'adhérence. Ensuite, nous avons comparé la similarité entre les résultats de *K-means* en utilisant les trois distances avec des centroïdes aléatoires et des centroïdes issus des sous-ensembles de fermées minimaux.

Nous avons utilisé la mesure de similarité proposée par [Torres 2009] qui permet d'évaluer la sortie du même algorithme de classification ou différents algorithmes de classification, exécutés sur le même jeu de données. Cette mesure a la particularité de comparer différents ensembles de partitions (*clusters* en anglais) sans faire appel à une base de vérité externe. Ceci fait qu'elle est adaptée à l'évaluation des algorithmes de classification (apprentissage non supervisé) où nous n'avons pas une base de vérité terrain pour évaluer notre algorithme, comme c'est le cas par exemple avec la discrimination (apprentissage supervisé).

Pour mesurer la similarité entre deux ensembles de partitions, nous utiliserons la fonction $Sim()$ telle qu'elle est présentée dans la définition 4.10.

Définition 4.10. Posons $C = \{C_1, C_2, \ldots, C_m\}$ et $D = \{D_1, D_2, \ldots, D_n\}$ comme étant le résultat de deux algorithmes de classification sur le même jeu de données. Supposons, que C et D sont deux algorithmes de classification «intolérants» ce qui veut dire que chaque élément du jeu de données ne peut appartenir qu'à une et une seule partition à la fois. Alors

$$S_{C,D} = \begin{bmatrix} S_{11} & S_{12} & S_{13} & \ldots & S_{1n} \\ S_{21} & S_{22} & S_{23} & \ldots & S_{2n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ S_{m1} & S_{m2} & S_{m3} & \ldots & S_{mn} \end{bmatrix} \tag{4.11}$$

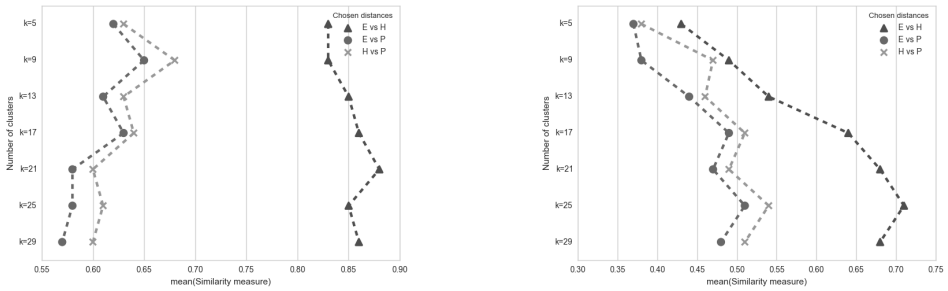
où $S_{ij} = \frac{p}{q}$ est le coefficient de similarité de Jaccard avec p la taille de l'intersection et a la taille de l'ensemble issue de l'union entre les deux ensembles de regroupements (*clusters*) C_i et D_j. La similarité entre le regroupement C des données et D est définie alors comme suit :

$$Sim(C, D) = \frac{\sum_{1 \leq i \leq m, 1 \leq i \leq m} S_{ij}}{\max(m, n)} \tag{4.12}$$

	Même algorithme			Mêmes centroïdes		
k	**E**	**H**	**P**	**E vs H**	**E vs P**	**H vs P**
5	0.423	0.454	0.381	0.838	0.623	0.631
9	0.487	0.544	0.423	0.831	0.665	0.684
13	0.567	0.598	0.405	0.855	0.615	0.633
17	0.645	0.658	0.419	0.861	0.630	0.641
21	0.676	0.707	0.445	0.880	0.581	0.604
25	0.736	0.720	0.452	0.856	0.583	0.613
29	0.723	0.714	0.442	0.864	0.578	0.600
moyenne	**0.608**	0.628	0.423	**0.855**	0.611	0.629

	Differents centroïdes			Centroides de l'adhérence		
k	**E vs H**	**E vs P**	**H vs P**	**E vs H**	**E vs P**	**H vs P**
5	0.434	0.373	0.383	-	-	-
9	0.495	0.383	0.447	-	-	-
13	0.546	0.445	0.469	**0.949**	**0.922**	**0.946**
17	0.641	0.493	0.518	-	-	-
21	0.687	0.478	0.491	-	-	-
25	0.715	0.519	0.540	-	-	-
29	0.684	0.4885	0.511	-	-	-
moyenne	**0.600**	0.454	0.480	**0.949**	**0.922**	**0.946**

TABLE 4.7 – Résultats de l'analyse de similarité entre les sorties des K-moyennes initialisés avec différentes mesures de distances. E, est la distance d'Euclide, H la distance de Hellinger, P est la distance de l'adhérence.



(*a*) (*b*)

FIGURE 4.4 – Illustration de la mesure de similarité avec les mêmes centroïdes initiaux (a) et différents centroïdes initiaux. E, est la distance d'Euclide, H la distance de Hellinger, P est la distance de l'adhérence.

4.1.4.5 Discussion

Nous avons comparé la similarité entre les résultats de trois algorithmes de classifications en utilisant un *K-means* avec différents centroïdes et différents nombres de partitions ou groupes k. Nous avons illustré les résultats de similarité entre les différentes partitions calculées avec les trois *K-means* dans les deux figures B.1 (a) et B.1 (b). Pour les résultats illustrés dans la figure B.1 (a), nous avons utilisé le même centroïde pour les 3 algorithmes de *K-means* avec la distance euclidienne, celle de Hellinger et celle que nous avons construite avec l'adhérence. Pour les résultats illustrés dans la figure B.1 (b), nous avons utilisé différents centroïdes pour les 3 algorithmes de *K-means*.

Dans les deux figures, nous avons noté que les algorithmes avec la distance euclidienne et celle de Hellinger ont tous deux une forte similarité. Ceci est dû au fait que les deux distances sont similaires. Dans la figure B.1 (a), nous voyons une grande séparation entre les partitions obtenues avec la distance construite avec l'adhérence, la distance euclidienne et celle de Hellinger. Cette séparation se réduit dans la figure B.1 (b) et augmente de nouveau à partir de $k = 17$. Une première conclusion nous incite à penser qu'avec différents centroïdes initiaux la distance construite avec l'adhérence a réduit la séparation entre les algorithmes de *K-means* utilisant la distance euclidienne et celle de Hellinger. Mais, quand $k > 13$, qui correspond au nombre des sous-ensembles de fermés minimaux, la séparation entre la distance de l'adhérence et les autres distances commencent à augmenter de nouveau.

Dans la table 4.7, nous avons rapporté les résultats des expérimentations, avec différentes configurations, que nous avons menées sur la similarité entre les partitions obtenues. La première conclusion qui ressort de ces résultats est l'importance du choix des centroïdes initiaux, différents centroïdes menant à des partitions finales complètement différentes. L'algorithme MCPTM offre une solution pour choisir les centroïdes à partir de la structure des données.

Les meilleurs résultats que nous avons obtenus sont : 94% de similarité entre la distance d'Euclide et Hellinger, 92% de similarité entre la distance d'Euclide et de l'adhérence, 94% de similarité entre la distance de Hellinger et de l'adhérence. Ces résultats ont été obtenus avec l'utilisation des centroïdes $\{0, 52, 4, 14, 26, 29, 30, 31, 34, 44, 85, 90, 99\}$ calculés avec notre algorithme MCPTM.

La contribution majeure de cette partie est de démontrer l'importance de l'extraction et de la sélection des caractéristiques pour la tâche de la classification et y apporter une solution. Notre méthode de classification multicritère associe l'analyse sémantique latente d'une collection de documents et la théorie de la prétopologie qui nous a permis de structurer l'espace des documents. L'allocation latente de Dirichlet est utilisée pour structurer les documents en des vecteurs de probabilités, et la fonction d'adhérence est utilisée pour relier ses vecteurs selon deux critères : les thèmes majeurs dans les documents, et la distance entre ces vecteurs. Nous avons appliqué notre approche sur les données issues du site de microblogging Twitter.

4.2 Gestion du passage à l'échelle

Dans la partie précédente, nous avons construit un espace sémantique de grande dimension pour modéliser les entités (i.e. documents) et leurs proximités sémantiques. Nous avons appliqué sur cet espace des modèles probabilistes pour l'identification et le suivi des structures latentes (i.e. thèmes). Nous avons travaillé avec l'un des modèles les plus utilisés pour l'identification des structures latentes appelé l'allocation latente de Dirichlet (LDA). Dans LDA chaque document est une distribution de thèmes représentée par la variable latente θ, chaque thème est une distribution de mots représentée par la variable latente ϕ et la probabilité qu'un thème soit relié à un mot dans le document est représentée par la variable latente z. Calculer les différentes variables latentes est un problème d'inférence bayésienne. Pour résoudre ce problème, deux des solutions les plus utilisées sont l'inférence variationnelle et l'échantillonnage de Gibbs. La première solution est plus rapide en exécution et la deuxième est plus lente, mais plus précise.

Pour l'identification et le suivi des structures thématiques, nous avons besoin de la rapidité d'exécution d'une part, mais également de la précision des résultats d'autre part. Néanmoins, le suivi dynamique de ces structures constitue un défi scientifique notamment en raison du passage à l'échelle lié au développement de l'espace sémantique au cours du temps. Face à des corpus de textes avec un volume considérable c'est-à-dire quelques milliers de documents, les itérations de l'échantillonnage de Gibbs prennent un temps conséquent qui se compte en jours et même en mois entiers [Newman 2009]. En réponse à ce problème, nous abordons dans cette section une solution de calcul parallèle en proposons une implémentation paralléle de l'échantillonnage de Gibbs.

La complexité algorithmique de l'échantillonnage de Gibbs est de l'ordre de $K \times W_{\mathcal{D}}$ [Mukherjee 2008], où K est le nombre de thèmes et $W_{\mathcal{D}}$ est le nombre d'occurrences des mots dans la collection de documents $\mathcal{D}$. Pour réduire cette complexité lors du passage à l'échelle, deux approches s'offrent à nous : la première approche consiste à réduire le volume de calculs en réduisant le volume de données à traiter ou en exploitant des caractéristiques particulières des variables utilisées lors du calcul. Par exemple, sachant que le vecteur de probabilités θ des thèmes dans un document est souvent creux (plusieurs 0) nous pouvons executer une réduction sur ce dernier pour accélerer le calcul comme dans les travaux de [Qiu 2014].

La deuxième approche consiste à utiliser des plateformes de calculs parallèles avec un réseau d'ordinateurs standard connectés en local et en haut débit. Au sein de cette approche nous avons d'abord un parallélisme des données sur les différentes unités, ensuite chaque processeur de chaque unité exécute le même algorithme sur les différentes données stockées en local. Au sein de cette approche, et dans le cadre de l'analyse catégorielle thématique des textes, un nombre de modèles de parallélisation synchrone est asynchrone sont proposés dans la littérature. [Nallapati 2007] ont présenté une exécution parallèle de l'algorithme espérance maximisation variationnelle, l'algorithme utilisé dans l'article originel de LDA pour l'inférence des variables latentes. [Newman 2009] ont proposé deux modèles synchrones pour distribuer l'algo-

rithme d'échantillonnage de Gibbs. Les deux méthodes sont appelées AD-LDA pour *Approximate Distributed LDA* en anglais ou parallélisation approximée de LDA, et HD-LDA pour *Hierarchical Distributed LDA* ou parallélisation hiérarchique de LDA. AD-LDA offre un modèle de parallélisation synchrone de l'échantillonnage de Gibbs réduit quand à HD-LDA elle offre un modèle de parallélisation synchrone de l'échantillonnage de Gibbs appliquer sur toute la structure des variables latente présente dans LDA. [Smyth 2009] ont présenté une parallélisation asynchrone de l'algorithme d'échantillonnage de Gibbs.

En pratique, l'utilisation des plateformes de calculs parallèles constitue un défi technique à cause notamment de la gestion des erreurs de calcul, la tolérance aux pannes/fautes et l'intégrité des données qui sont distribuées sur les différentes unités du réseau [Tanenbaum 1994]. Des espace de travail comme *MapReduce* [Dean 2008] ou encore MPI pour *Message Passing Interface* [Thakur 2005] nous offre un mécanisme d'exécution qui prend en charge les défis cités. Ils nous permettent donc de nous concentrer uniquement sur la parallélisation de l'algorithme qui nous intéresse, dans ce cas l'algorithme de Gibbs.

Par exemple, [Wang 2009] ont utilisé MPI ou le passage de message entre les différentes partitions sur lesquelles les implémentations de Gibbs s'exécutaient en local, pour s'assurer de la consistance de leurs résultats. Par manque d'un système de tolérance aux fautes, le nombre de lectures/écritures des messages entre les paritions leur a coûté cher en temps et vitesse d'exécution.

Contrairement à MPI, MapReduce implémente la tolérance aux fautes en partitionnant le calcul en grains fins et traçables. De sorte que s'il ya une panne sur une unité ou noeud du réseau, le système reconduit les grains de calculs sur un autre noeud. Hadoop est une plateforme logicielle qui implémente le mécanisme de parallélisation MapReduce.

Par exemple, [Smola 2010] ont utilisé Hadoop pour implémenter LDA. Cependant, bien que Hadoop offre une gestion efficace de parallélisation des données il n'est pas adapté à la parallélisation d'algorithmes itératifs comme l'échantillonage de Gibbs et ce pour deux raisons. La première est la façon dont est conçu le système de tolérance aux fautes de MapReduce qui impose le stockage du regroupement des résultats de chaque noeud après exécution. Néanmoins, ces résultats ne sont qu'intérmédiaire dans le cas d'un algorithme comme celui de l'échantillonage de Gibbs où il s'agit de converger vers une distribution de probabilité après plusieurs itérations. Le stockage de ces résultats sucite des écritures sur le disque qui résulent en des pertes de performances. La deuxième est le manque de générateur de nombres aléatoires directement implémenté dans Hadoop. Une implémentation réalisée par [Ming 2014] a comblé ce manque mais ceci rajoute une couche de complexité supplémentaire à l'implémentation totale.

Spark est une plateforme logicielle pour le calcul parallèle [Zaharia 2010]. La solution Spark est orientée vers le traitement intensif des données à grande dimension comme les données textuelles. Les résultats intérmédiaires sont conserver sur la mémoire vive. Dans le cas d'une panne d'un noeud un historique est rechargé permettant de retracé la opérations qui ont permis d'obtenir les données perdues.

Ce système permet de garder la tolérance aux fautes de MapReduce et permet d'obtenir une accélaration 10 fois plus importante pour les algorithmes itératifs grace à l'utilisation de la mémoire vive.

A notre connaissance, il existe deux travaux similaires à notre implémentation avec Spark de l'échantilloange de Gibbs appliquée à LDA. Le premier [Qiu 2014] ne donne pas de détails par rapport à la synchronisation globale des différentes variables aléatoires calculées sur les processeurs. Mais, la différence principale avec notre proposition c'est la non-synchronisation de la matrice d'occurrence des thèmes dans les documents, dans leur implémentation. Nous considérons que la synchronisation de cette matrice est importante pour décider du nombre d'itérations lors de la phase d'échantillonnage (e.g. quand la valeur de la matrice ne change plus avec les itérations). Le deuxième travail [mer 2015] est une proposition de code, mais l'auteur ne donne pas plus de détails sur son implémentation ni les performances de cette dernière.

Notre contribution se résume dans les points suivants :

1. Nous proposons une implémentation parallèle de l'algorithme de Gibbs réduit basée sur le modèle synchrone de [Newman 2009]. Notre version construite avec la syntaxe Spark, permet un traitement rapide des données ainsi qu'une synchronisation efficace des variables latentes du modèle sur les différentes machines de la plateforme de calcul parallèle.
2. Nous démontrons avec un ensemble d'expérimentations que notre version distribuée et la version séquentielle tournant sur un seul CPU ont le même pouvoir prédictif.
3. Nous démontrons la capacité de déploiement de notre version distribuée sur une large grappe d'ordinateurs avec une analyse de l'accélération gagnée.

L'organisation de cette section est comme suit : dans la première sous-section, nous présentons le modèle synchrone sur lequel nous avons fondé notre implémentation Spark. Nous détaillons ensuite, notre implémentation et nous présentons enfin les résultats obtenus lors du passage à l'échelle en utilisant des corpus de données de grande taille.

4.2.1 Modèle synchrone pour la parallélisation de LDA

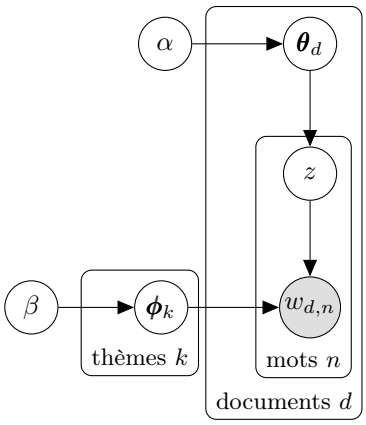


FIGURE 4.5 – Illustration du réseau bayésien de l'allocation latente de Dirichlet, les variables observées sont en gris, ici les mots $w_{d,n}$.

Nous souhaitons distribuer le modèle de l'allocation latente de Dirichlet décrit par le modèle graphique probabiliste (PGM pour *Probabilistic Graphical Model*). Le PGM de LDA illustré dans la figure 4.5 représente un réseau bayésien. À partir de ce réseau bayésien, nous pouvons écrire l'équation 4.13 de Bayes.

$$p(\theta, \phi, z|w, \alpha, \beta) = \frac{p(\theta, \phi, z, w|\alpha, \beta)}{p(w, \alpha, \beta)} \tag{4.13}$$

Un calcul exacte de l'*a posteriori* de cette dernière n'était pas calculable en temps polynomial à cause de son facteur de normalisation [Sontag 2011]. Pour résoudre ce problème nous avons décidé d'utiliser un algorithme de l'échantillonnage de Gibbs pour l'approximation de l'*a posteriori*.

Notre implémentation parallèle de l'algorithme de Gibbs appliqué à LDA, est fondée sur les travaux de [Newman 2009] et leur modèle synchrone appelé AD-LDA. Dans AD-LDA, la collection de documents D est devisée sur P processeurs avec $\frac{D}{P}$ documents sur chaque processeur approximativement. L'algorithme de l'échantillonnage de Gibbs est par la suite codé sur chaque processeur et exécuté simultanément sur $\frac{D}{P}$ documents pour approcher une nouvelle valeur de z_i avec l'équation 4.14. Cette equation représente la probabilité que le thème j est choisi pour un mot w_i étant donné tous les thèmes $-i$ choisi pour tous les mots dans le document d_i.

$$P(z_i = j \mid z_{-i}, w_i) \propto \frac{n_{-i,j}^{w_i} + \beta}{n_{-i,j}^{(\cdot)} + V\beta} \frac{n_{-i,j}^{d_i} + \alpha}{n_{-i,\cdot}^{d_i} + K\alpha} \tag{4.14}$$

Où $n_{-i,j}^{w_i}$ est le nombre d'occurrences du mot w_i dans le thème j. $n_{-i,j}^{(\cdot)}$ est le nombre d'occurrences de tous les autres mots dans le thème j. $n_{-i,j}^{d_i}$ est le nombre d'occurrences du thème j dans le document d_i. Enfin, $n_{-i,\cdot}^{d_i}$ est le nombre d'occurrences de tous les autres thèmes dans le document d_i.

En pratique les différentes occurrences sont représentées dans des matrices. À chaque itération nous calculons la probabilité $P(z_i = j \mid z_{-i}, w_i)$, c'est à dire la probabilité que le thème j soit affecté au mot w_i conditionné par tous les autres thèmes $-i$. Si cette probabilité est plus grande pour le thème j que les autres thèmes $-i$ alors on recompte les occurrences $n^{w_i}_{-i,j}$ $n^{(\cdot)}_{-i,j}$ $n^{d_i}_{-i,j}$ et $n^{d_i}_{-i,\cdot}$. Ce sont ces opérations de calcul de probabilités et de comptage que nous distribuons sur chaque processeur. Les occurences $n^{w_i}_{-i,j}$ et $n^{(\cdot)}_{-i,j}$ vont en paire si on augumente la valeur de l'une il faut diminuer la valeur de l'autre, idem pour $n^{d_i}_{-i,j}$ et $n^{d_i}_{-i,\cdot}$. Nous noterons la première paire n^{wj} et la deuxième n^{dj} et nous y référons comme des comptes communs.

Chaque processeur p après l'exécution d'une seule itération de l'algorithme de Gibbs et le calcul de la probabilité $P(z_i = j \mid z_{-i}, w_i)$ possède ses propres valeurs de n_p^{wj} et n_p^{dj}. Une fois l'itération terminée chaque processeur p doit remonter les résultats qu'il a obtenus. C'est-à-dire le nombre d'occurrences du mot w dans le thème j et le nombre d'occurrences du thème j dans le document d, des valeurs que nous trouvons respectivement dans n_p^{wj} et n_p^{dj}. Tous les résultats sont rassemblés et des valeurs communes que nous dénoterons n^{wj}_{new} et n^{dj}_{new} seront mis à disposition pour tous les processeurs, et ce pour la prochaine itération. Nous parlerons aussi de réduction de tous les n_p^{wj} et n_p^{dj} en des comptes communs à tous les processeurs. n^{wj}_{new} et n^{dj}_{new} sont calculés respectivement avec l'équation et .

$$n^{wj}_{\text{new}} = n^{wj}_{old} + \sum_p (n^{wj}_{\text{p}} - n^{wj}_{old}) \tag{4.15}$$

$$= \sum_p n_p^{wj} - (p-1) n^{wj}_{old} \tag{4.16}$$

où n^{wj}_{old} est la valeur de l'occurence du mot w dans le thème j commune à tous les processeurs et calculée dans l'itération précédente.

$$n^{dj}_{\text{new}} = n^{dj}_{old} + \sum_p (n^{dj}_{\text{p}} - n^{dj}_{old}) \tag{4.17}$$

$$= \sum_p n_p^{dj} - (p-1) n^{dj}_{old} \tag{4.18}$$

où n^{dj}_{old} est la valeur de l'occurence du thème j dans le document d commune à tous les processuers et calculée dans l'itération précédente.

Les deux inconvénients majeurs de AD-LDA sont : (i) AD-LDA nécessite de garder une copie de tous les comptes communs, n^{wj} et n^{dj}, sur les P processeurs. Ceci peut devenir problématique et résulter en des perturbations lors de l'exécution de l'algorithme sur des jeux de données de taille importante. (ii) La façon avec laquelle sont calculées n^{wj} et n^{dj} peut résulter en des conflits d'écriture/lecture sur la mémoire, ce qui affectera la justesse des résultats.

Pour résoudre ces problèmes, nous avons choisi de proposer une extension de l'algorithme AD-LDA sous la forme d'une implémentation utilisant une syntaxe de programmation fonctionnelle supporté par la plateforme de calcul parallèle appelée Spark. Spark fournit un calcul en mémoire vive, ce qui fait que nous n'avons pas besoin de garder les valeurs de n^{wj} et n^{dj}. Nous pouvons également éviter les conflits de lecture/écriture sur la mémoire en diffusant à tous les processeurs une copie temporaire de n^{wj} et n^{dj}.

4.2.2 Implémentation parallèle de LDA avec Spark

La syntaxe Spark est une généralisation du mécanisme *MapReduce*, où nous avons deux suites d'instructions principales à travers lesquelles nous pouvons distribuer le calcul. La première suite est représenter par la fonction *Map* et la deuxième par la fonction *Reduce*. La collection de textes passe d'abord par une ou plusieurs fonctions *Map*, qui y appliquent certaines tâches. Ensuite, une ou plusieurs fonctions *Reduce* rassemblent la sortie de chaque tâches pour calculer un résultat global à un groupe de processeurs ou à tous les processeur du système parallélisé.

Une ou plusieurs fonctions *Map* et/ou *Reduce* peuvent être affecté à un ou plusieurs noeud (un ordinateur, ou un processeur multi-coeur) du réseau de la plateforme de clacul parallèle. L'exécution de chacune des deux fonctions ne suit pas un certain ordre précis.

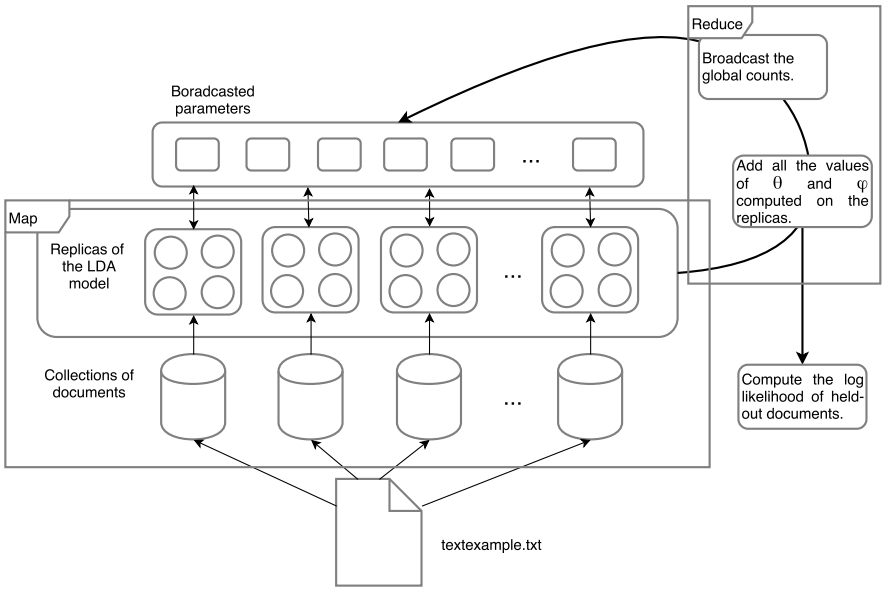


FIGURE 4.6 – Flux de travail de l'implémentation Spark. Chaque itération est exécutée dans l'espace de travail *Map* et *Reduce*. Le modèle LDA est copié sur les p processeurs, et le document de travail fourni en entrée est divisé en p ensemble de documents. Après l'application de la procédure *Reduce* les paramètres globaux n^{wj} et n^{dj} sont diffuser à l'ensemble des répliques de LDA. Le logarithme de la vraisemblance est calculé sur l'ensemble des documents tests après chaque itération.

Notre implémentation parallèle de LDA avec Spark est décrite par le fonctionnement suivant :

1. La collection de textes est rassemblé dans un seul fichier unicode. L'ensemble des données textuelles est dévisé en fragments.
2. Une ou plusieurs fonctions *Map* sont affectées à chaque noeud qui reçoit un ou plusieurs fraguement de données textuelles pour lesquelles il génére une sorte d' index avec une clé et une valeur. Cette étape permet d'associer à chaque noeud à travers une clé donnée une fonction *Reduce*.
3. Chaque noeud initialise les différentes variables latentes selon le modèle LDA, ensuite exécute l'algorithme de Gibbs sur le ou les fragments qui lui sont associés et ce pour approximer chaque variable latente.
4. Les index produits par les fonctions *Map* sont regroupés par clés et les fonctions *Reduce* reprennent les résultats des traitements effecutés sur chaque fragements qui correspond à leur clé respective.
5. Chaque fonction *Reduce* combine les résultats des *Map* selon la clé qui lui est associée, stocke le résultat dans la mémoire vive, et partage le résultat sur les différents noeuds pour d'autres itérations.

Ce fonctionnement est illustré dans la figure 4.6. La suite d'instructions de la fonction *Map* est décrite dans l'algorithme 7, elle représente les instructions exécutées sur chaque noeud ou processeur. La suite d'instruction de la fonction *Reduce* est décrite dans l'algorithme 8, elle représente les instructions nécessaires pour d'abord combiner les résultats de chaque processeur, et ensuite les communiquer de nouveau à tous les processeurs.

Algorithme 7 Map

Précondition : P processeur ou noeuds pour la collection des documents D

pour tout partitions **faire**
  $n_p^{wj} = n^{wj}$
  $n_p^{dj} = n^{dj}$
  pour tout $w \in d_i$, $d_i \in D$ et z_i dans z **faire**
    Echantillonner z_{ij} à partir de n_p^{wj} et n_p^{dj} (voir équation 4.14)
    Calculer $n_{p_{new}}^{wj}$ et $n_{p_{new}}^{dj}$ à partir de z_{ij}.
  Calculer $\hat{\phi}_j^{(w)}$ et $\hat{\theta}_j^d$ avec les équations 4.19 et 4.20.

Dans l'algorithme 7, les instructions de la fonction *Map* sont exécutées sur les P processeurs. La fonction *Map* prend en entrée un fragement de la collection de document D et nous retourne en sortie la distribution des thémes dans les documents présents sur le noeud et la distribution des mots dans chacun de ces thèmes. Les deux premières instructions sont des instructions d'initialisation des comptes commun n_p^{wj} et n_p^{dj} avec les valeurs obtenues à la dernière itération. Si c'est la première itération, nous les initialisons à zéro. Nous calculons ensuite le thème z_{ij} et produisons de

nouveaux comptes communs pour calculer la distribution θ des thèmes par document et la distribution ϕ des mots par thème avec les équations 4.20 et 4.19.

$$\hat{\phi}_j^{(w)} = \frac{n_j^{(w)} + \beta}{n_j^{(\cdot)} + V\beta} \tag{4.19}$$

$$\hat{\theta}_j^{(d)} = \frac{n_j^{(d)} + \alpha}{n_\cdot^{(d)} + K\alpha} \tag{4.20}$$

Algorithme 8 Reduce

1: Réduire toutes les instances $n_{p_{new}}^{wj}$ avec l'équation 4.15.
2: Réduire toutes les instances $n_{p_{new}}^{dj}$ avec l'équation 4.17.
3: Distribuer les résultats $n_{p_{new}}^{wj}$ et $n_{p_{new}}^{dj}$ sur toutes les instances p.
4: Ajouter à θ toutes les θ_p
5: Ajouter à ϕ toutes les ϕ_p

L'algorithme 8 est exécuté à la fin de chaque itération de l'échantillonnage de Gibbs. Ceci afin de combiner les résultats des P processeurs ou noeuds, c'est-à-dire les résultats obtenus en sortie des fonctions *Map* déployées sur chaque noeud. L'algorithme 8 combine d'abord les comptes communs $n_{p_{new}}^{wj}$ et $n_{p_{new}}^{dj}$.C'est-à-dire le nombre d'occurrences du mot w dans le thème j et le nombre d'occurrences du thème j dans le document d et cela pour chaque processeur p. Ensuite, les résultats combinés sont calculés et enregistrés respectivement dans $n_{p_{new}}^{wj}$ et $n_{p_{new}}^{dj}$. L'algorithme 8, distribue ensuite $n_{p_{new}}^{wj}$ et $n_{p_{new}}^{dj}$ à tous les noeuds ou processeurs pour que les valeurs soient utilisées au début de la prochaine itération. Enfin, l'algorithme 8 combine les valeurs de probabilités de la distribution de thèmes θ dans les documents et les valeurs de probabilités de la distribution ϕ des mots dans les thèmes. Avant la convergence de l'algorithme de Gibbs, les valeurs de cet ensemble de variables sont intermédiaires et ils sont stockés dans la mémoire vive. Les algorithmes 7, 8 sont exécutées jusqu'à convergence de l'échantillonnage de Gibbs ou jusqu'à ce que la valeur de $n_{p_{new}}^{dj}$ ne change plus avec les itérations.

Algorithme 9 Version distribuée de LDA

Précondition : w mots dans le corpus de document $D = (d_1, d_2, \ldots, d_n)$
1: **si** première itération **alors**
2: Initialiser aléatoirement les comptes communs
3: **pour tout** itérations **faire**
4: Distribuer le fichier sur les RDDs avec l'algorithme 7.
5: Calculer les comptes communs avec l'algorithme.
6: Distribuer les comptes communs sur les RDDs.
7: Calculer sur chaque RDD $\hat{\phi}_j^{(w)}$ et $\hat{\theta}_j^d$.
8: Récupérer à partir de chaque RDD les valeurs de $\hat{\phi}_j^{(w)}$ et $\hat{\theta}_j^d$ avec l'algorithme 8
9: **retour** z, $\hat{\phi}_j^{(w)}$ et $\hat{\theta}_j^d$

La principale nouveauté qu'apporte Spark est le concept des RDD (pour *Resilient Distributed Dataset*. Les RDDs contiennent la collection de documents partitionnés

en fragments et distribués sur les noeuds par la fonction *Map*. Les collections RDDs sont conservées en mémoire vive (RAM), la résilience des données est assurée par la conservation des lignes de code qui ont construit ces fragments. Le programme qui contient la fonction *Map* et la fonction *Reduce* est appelé programme pilote. Il est décrit dans l'algorithme 9, où nous avons les instructions pour distribuer la collection de documents D sur les RDDs disponibles, et l'exécution des algorithmes 7 et 8.

4.2.3 Application aux données et évaluation

	Nips	**KOS**	**NYT**
D	1,500	3430	300,000
V	12,419	6906	102,660
N	2,166,058	467714	99,542,125

TABLE 4.8 – Description des quatre jeux de données utilisés pour l'évaluation du passage à l'échelle.

Pour évaluer notre version distribuée, nous avons choisi trois jeux de données fournis par le dépôt de l'université de Californie à Irvine. La table 4.8 résume les caractéristiques des jeux de données où D est le nombre de document, V est la taille du vocabulaire, et N est le nombre de mots. Nous considérons Nips et KOS comme deux jeux de données à dimension réduite, nous les avons utilisés pour rapporter la perplexité entre la version séquentielle et la version distribuée. Nous considérons NYT comme un jeu de données à grande dimension, nous l'avons utilisé pour rapporter les expérimentations sur l'accélération des performances de l'algorithme et son comportement face au passage à l'échelle.

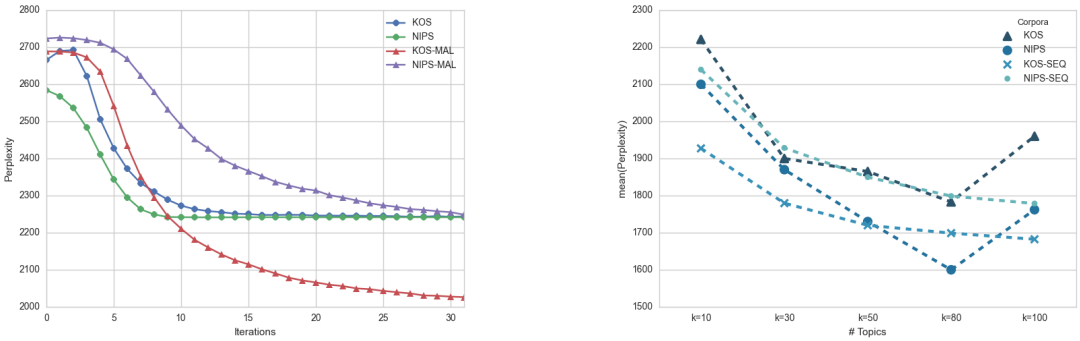
Les expérimentations que nous avons menées pour évaluer notre implémentation de la version distribuée de LDA sont : (i) la perplexité ou la vraisemblance des documents mis à l'écart pour le test. (ii) Le gain de vitesse par rapport à la version séquentielle exécutée sur un seul processeur. (iii) La scalabilité ou l'extensibilité de notre implémentation et son comportement lors de son déploiement sur plusieurs processeurs.

La perplexité est une mesure d'évaluation de la qualité de prédiction d'un LDA entraîné. On mesure le logarithme de la vraisemblance des documents mis à côté, c'est-à-dire qui ne sont pas utilisés lors de l'entraînement de LDA. La perplexité est définie comme suit.

$$p(x^{test}) = \prod_{ij} \log \frac{1}{S} \sum_{s} \sum_{k} \hat{\theta}^{s}_{jk} \hat{\phi}^{s}_{x_{ij}k} \tag{4.21}$$

où N_{test} est le nombre de documents de test et $p(x^{test})$ est la vraisemblance calculée sur les mots W_d du document d.

Pour évaluer la justesse de notre implémentation distribuée de LDA avec la perplexité, nous avons utilisé pour chaque jeu de données, 75% des documents pour



(a) (b)

FIGURE 4.7 – Illustration de la valeur de la perplexité par rapport au nombre d'itérations dans la figure (a) et par rapport au nombre de thèmes dans la figure (b). Dans la figure (a) nous comparons notre version distribuée avec la version distribuée de MALLET [McCallum 2002]. Dans la figure (b) nous comparons la perplexité de notre version distribuée avec une version séquentielle.

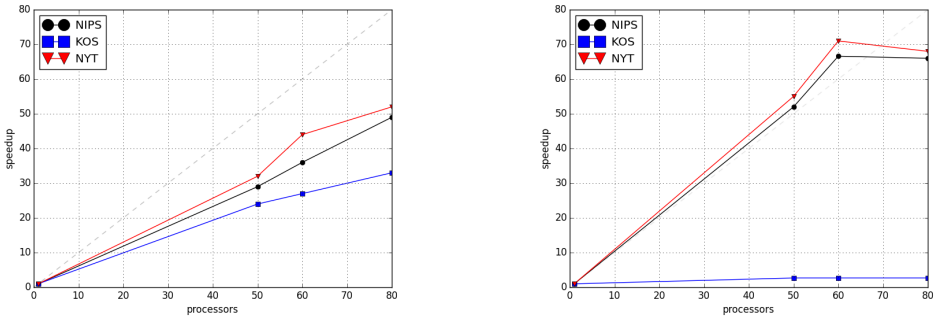
l'entraînement et 25% des documents pour les tests. Dans la figure (a) 4.8 nous comparons la valeur de la perplexité obtenue avec une version séquentielle et celle obtenue avec notre implémentation distribuée sur Spark et ce pour différents nombres de thèmes. Nous observons que notre implémentation a le même pouvoir prédictif que la version séquentielle. Nous avons obtenu pour le jeu de données NIPS, qui est plus important en taille que KOS, une meilleure valeur de perplexité. Par exemple, quand $K = 80$, nous obtenons une perplexité de 1600 comparée à la version séquentielle qui obtient une perplexité égale à 1800.

Pour un nombre fixe de K, nous observons dans la figure (b) 4.8 que notre implémentation converge vers des valeurs similaires à une version standard de LDA. Dans cette figure, nous comparons notre version à la version distribuée de MALLET [2] [McCallum 2002] qui est l'une des versions les plus utilisées pour l'identification des thèmes. Mallet [McCallum 2002] utilise du *multithreading* pour distribué le calcule de l'échantillonnage de Gibbs. Le *multithreading* consiste à exécuter plusieurs tâches en simultannée sur un seul processeur. l'éxécution se déroule en parallèle avec un partage d'une mémoire virtuelle. Nous avons utilisé pour comparer l'implémentation pralléle de MALLET et notre implémentation SPARK les paramètres suivant : 12 threads pour Mallet, 12 processeurs pour notre version Spark et un nombre de thèmes K = 10.

Notons que dans la figure (b) 4.8, notre implémentation sur Spark représenté par la courbe en points ronds converge avant l'instance distribuée de Mallet représenté par la courbe avec des points en triangle. Et ceci, pour les deux jeux de données NIPS et KOS. Pour le jeu de données le plus petit, c'est-à-dire KOS, Mallet semble

2. MALLET est un espace de travail pour l'analyse catégorielle des textes. MALLET offre une implémentation séquentielle de LDA ainsi qu'une version distribuée. http://mallet.cs.umass.edu

avec un meilleur pouvoir prédictif. Cependant, nous obtenons de meilleurs résultats sur le jeu de données NIPS avec un taux de convergence plus élevé.



(a) (b)

FIGURE 4.8 – Illustration du gain de vitesse (a) par rapport aux nombres de processeurs (b) Illustration du gain de vitesse en tenant compte du passage à l'échelle.

Nous rapportons les deux mesures suivantes : la première, illustrée dans la figure (a) 4.8, est l'accélération gagnée en exécutant l'algorithme de Gibbs réduit en utilisant notre implémentation sur Spark. La deuxième, illustrée dans la figure (b) 4.8, est une métrique relative au comportement de l'implémentation lors du passage à l'échelle. La première mesure est une comparaison de la vitesse gagnée par une implémentation sur deux architectures avec différentes ressources. Nous avons calculé cette dernière avec la loi de [Amdahl 1967] présentée dans l'équation suivante :

$$S = \frac{1}{1 - p + \frac{p}{s}} \tag{4.22}$$

Où p est le pourcentage ou la portion, du processus entier, qui a bénéficié d'un changement dans les ressources de l'architecture sur laquelle il s'exécute. s est le nombre rendu disponible dans la nouvelle architecture. Dans notre cas, s est égale au nombre de processeurs. Nos expérimentations sont menées sur les jeux de données NIPS, KOS, et NYT.

Le processus entier qui a bénéficié de la distribution est l'algorithme de l'échantillonnage de Gibbs réduit, nous dénoterons ce processus par *pgr* pour processus de Gibbs réduit. En particulier, et comme c'est montré dans l'algorithme 8, nous avons trois portions qui divisent l'exécution du *pgr* : d'abord, réduction de toutes les instances de $n^{wj}_{p_{new}}$. Ensuite, réduction de toutes les instances de $n^{dj}_{p_{new}}$. Enfin Distribution des résultats $n^{wj}_{p_{new}}$ et $n^{dj}_{p_{new}}$ sur toutes les instances p. L'accélération totale est calculée comme suit, $S = \frac{1}{\frac{p1}{s1} + \frac{p2}{s2} + \frac{p3}{s3}}$.

Nous pouvons observer dans la figure (a) 4.8, une forte corrélation entre l'augmentation de la taille du jeu de données utilisé et l'accélération du LDA Spark par rapport à la version non distribué. Pour le jeu de données KOS, nous observons une accélération qui arrive à une constante à partir de 60 processeurs. Ceci est dû au petit nombre de mots distribué dans les partitions et la distribution finit par n'avoir aucun effet sur le temps de l'échantillonnage.

Une fois que nous avons implémenté un algorithme avec la syntaxe de Spark et avons eu de bons résultats sur un petit nombre de noeuds sur une grappe d'ordinateurs, nous pouvons rajouter un nombre arbitraire de noeuds sans aucun effort supplémentaire de développement. Nous nous référons ici à la capacité de l'algorithme ou de l'implémentation à passer à une échelle plus importante de données et de ressources dans l'architecture sur laquelle elle s'exécute. La deuxième mesure dans cette partie de l'évaluation de notre implémentation Spark est relative à la capacité de passer à une échelle plus importante. Pour calculer cette métrique, nous utiliserons la loi universelle du passage à l'échelle introduite par [Gunther 2008], pour analyser la configuration du système idéal que permet d'avoir une accélération optimale.

L'accélération calculée avec l'équation 4.22, est relative au processus de l'échantillonnage pgr. Nous voulons ici calculer la scalabilité de l'implémentation entière de l'allocation latente de Dirichlet avec la syntaxe Spark. D'abord, nous reprenons la définition de l'accélération comme suit :

$$S_p = \frac{T_1}{T_p} \tag{4.23}$$

Où T_1 est le temps d'exécution mesuré sur un processeur et T_p est le temps d'exécution mesuré sur p processeurs. L'équation 4.23 est une généralisation de la loi de Amdahl. Elle est obtenue à partir de l'équation suivante :

$$S_p = \frac{p}{1 + \sigma(p-1) + \kappa p(p-1)} \tag{4.24}$$

Où σ est le degré de contention entre les différentes parties du système et κ représente le temps de latence nécessaire pour garder la cohérence du système. On parle de contention quand deux tâches au sein du même noeud sont en concurrence pour accéder à un fragment sur le RDD.

Dans la figure (b) 4.8, nous avons illustré le comportement de l'implémentation Spark par rapport à la scalabilité. Pour le jeu de données KOS, nous observons un comportement sublinéaire de l'accélération . Nous pouvons conclure que pour le jeu de données KOS, l'implémentation Spark n'apporte pas d'accélération significative par rapport à la version séquentielle. Pour le jeu de données NIPS, nous observons un comportement sur-linéaire de l'accélération. À partir de 65 processeurs, la courbe de la scalabilité du NIPS coupe la limite linéaire et rentre dans ce que l'on appelle la région de remboursement (*payback region* en anglais). Ce comportement nous rappelle que les performances d'une plateforme de calcul parallèle nécessitent parfois de faire des compromis. Par exemple, ceci revient à faire des compromis sur le nombre de noeuds, car le temps de latence nécessaire pour garder la cohérence du système ralentit la performance totale. Où encore faire des compris sur le nombre de fragments, car le degré de contention cause plusieurs conflits qui finissent aussi par ralentir le système. Dans ce contexte, une étude détaillée a été menée par [Gunther 2015] sur le mécanisme MapReduce et les compris à faire.

Dans ce chapitre, nous avons traité la problématique de la classification et de la discrimination des documents en tenant compte des thèmes qu'ils relatent. Nous avons approché le problème d'abord, en construisant un espace sémantique latent en utilisant la méthode de l'allocation latente de Dirichlet puis, en exploitons la notion de "voisinage" en proposons des mesures de proximités entres les entités dans ces espaces.

Nous avons proposé deux méthodes de classification. La première permet de regrouper les documents selon deux critères : le premier est qualitatif ; il est représenté par les thèmes majeurs dans la collection des documents. Le deuxième est quantitatif ; il est représenté la distance entre la distribution des thèmes dans chaque document de la collection. Au sein de cette méthode, nous avons procédé à l'analyse de la structure de l'espace sémantique latent à l'aide de la prétopologie. Ceci nous a permis de fournir, à la première méthode de classification, le nombre de classes ou groupes ainsi que leurs centroïdes respectifs. Ceci nous une initialisation aléatoire de ces paramètres et permet d'obtenir une meilleure qualité dans les regroupements de documents.

Nous avons également proposé dans cette partie, une version distribuée de l'allocation latente de Dirichlet avec une parallélisation de l'algorithme de l'échantillonnage de Gibbs. Ceci afin d'assurer la bonne gestion du passage à l'échelle des deux méthodes que nous avons proposées après.

Chapitre 5

Classification hiérarchique des pages web avec une analyse thématique catégorielle

Résumé : L'allocation latente de Dirichlet (LDA), avec l'application de l'échantillonnage de Gibbs, peut être considérée comme un algorithme d'apprentissage non supervisé des thèmes dans les textes. LDA représente les thèmes par une distribution de mots, chaque mot possède différentes valeurs de probabilités selon le thème. L'affectation d'une étiquette thématique à une distribution de mots est une étape manuelle et subjective. Afin de contourner ce problème et orienter l'algorithme vers l'apprentissage automatique d'une catégorie de thèmes prédéfinis, nous avons développé une version semi-supervisée de LDA. Ceci nous a servi par la suite pour entraîner des forêts aléatoires afin de classifier les documents selon une hiérarchie de thèmes en partant du thème le plus général vers le thème le plus particulier. Ce type d'approche est utile pour l'archivage automatique d'une grande collection de documents.

Avec le développement de l'internet et des réseaux sociaux, nous constatons une augmentation de la production de pages web et de la production d'articles textuels en tout genre. La classification des pages web est une tâche importante et cela pour plusieurs raisons. Premièrement, elle permet de développer et d'assurer la maintenance automatique des archives ou des répertoires de ces pages. Deuxièmement, elle permet aux moteurs de recherches de fournir plusieurs listes ordonnées en réponse à une requête donnée où chaque liste correspond à une catégorie en particulier. Cela peut aider notamment l'utilisateur à avoir un plus large spectre sur les résultats qui lui sont retournés, ou encore cela peut l'aider à repréciser sa requête. Troisièmement, la récupération des données textuelles depuis le web, connue également sous le nom de *crawling* en anglais, est une tâche compliquée à cause de l'hétérogéinité des sources de données sur le web.

En réponse à ces différents besoins, nous adressons dans ce chapitre une nouvelle méthode de classification souple et hiérarchique des textes. Nous qualifions cette méthode de souple, car un document peut appartenir à une ou plusieurs catégories. Nous la qualifions de hiérarchique, car les catégories auxquelles un document appartient sont hiérarchisées. Nous souhaitons avec notre méthode classifier un document donné à travers une hiérarchie de catégories, en partant de la plus générale vers la plus particulière.

Ce problème est abordé dans la littérature de différentes manières. Par exemple,

[Patil 2012] ont utilisé un modèle de langage avec une classification naïve bayésienne des pages. [Klassen 2010] ont utilisé les forêts aléatoires avec une sélection de mots-clés extraite des documents pour entraîner ces derniers. D'autres caractéristiques ont été utilisé par [Liparas 2014] pour entraîner également des forêts aléatoires. Les auteurs se sont référées à ces caractéristiques comme étant multimodales, c'est-à-dire textuelles et visuelles.

Le principal défi lors de l'entraînement des forêts aléatoires est l'extraction de caractéristiques textuelles adaptées pour la tâche que nous intéresse. [Liparas 2014] ont utilisé des n-grammes, quant à [Klassen 2010], ils ont rajouté une phase de sélection des n-grammes en calculant la fréquence des mots. Dans les deux approches la classification est basée sur l'extraction ensuite la sélection d'un nombre de mots clés. Ces mots clés sont affectés à des catégories, ensuite l'algorithme de classification discrimine le contenu textuel des pages web par rapport à ces mots clés.

Nous utilisons également les forêts aléatoires pour la classification des pages web. Néanmoins notre approche est différente, car l'algorithme de classification discrimine les pages web non pas directement selon leur contenu textuel, mais selon la distribution de thèmes de chaque page web. Cette approche nous permet d'offrir une navigation dans les pages web par thèmes et sous thèmes au lieu d'une navigation par mots clés. Elle nous permet également de construire automatiquement des archives avec des hiérarchies de thèmes partant des plus généraux vers les plus particuliers où une page web peut appartenir à plusieurs niveaux dans la hiérarchie en même temps.

Pour extraire les distributions de thèmes à partir des pages web, nous utilisons l'allocation latente de Dirichlet (LDA). LDA avec l'application de l'échantillonnage de Gibbs peut être considéré comme un algorithme d'apprentissage non supervisé qui ne nous donne pas la possibilité d'extraire un ensemble de thèmes prédéfinis avec une certaine étiquette. C'est pour cette raison, nous avons développé une version semi-supervisée tirée des travaux de [Ramage 2009, Lu 2013].

Nous nous référons à cette version comme étant semi supervisé, car nous gardons les caractéristiques d'un apprentissage non supervisé, mais nous orientons cet apprentissage par un certain nombre d'interventions. Par exemple, nous allons préférer un résultat d'un échantillonnage par rapport à un autre si les mots dans la distribution de thèmes échantillonnés contiennent certains mots qui appartiennent à une catégorie en particulier.

Ce chapitre est organisée comme suit. Nous présentons d'abord la version semi-supervisée de LDA, nous abordons ensuite notre modèle de travail fondé sur l'utilisation des forêts aléatoire. Nous présentons enfin l'application de notre méthode et son évaluation sur les données récupérées du répertoire de pages web dmoz. L'évaluation que nous avons menée explore plusieurs stratégies qui correspondent à la sélection de différentes caractéristiques telles que les différents niveaux de catégories ou le nombre d'arbres dans les forêts aléatoires.

5.1 Version semi-supervisée de LDA

La supervision du processus d'identification des thèmes se fait sur deux niveaux. Elle se fait d'abord au niveau des mots dans les documents, ensuite au niveau des documents. D'abord, pour chaque mot d'un document d présent dans la liste des mots clés $\Omega = \omega_1, \omega_1, \ldots, \omega_{n_{\mathbb{KV}}}$, nous affectons à ce mot un thème cible appartenant à une liste restrictive de thèmes cibles $T_{i,j} = \{T_{i,j_1}, T_{i,j_2}, \ldots, T_{i,j_{\mathbb{K}}}\}$. Nous obtenons, au terme de cette étape, un dictionnaire que l'on annotera $\mathbb{W}_t$ où nous avons pour chaque mot clé ω_i une liste de thèmes qui lui sont associés $W_t[\omega_i]$.

Ensuite, au niveau des documents, nous étiquetons avec un ou plusieurs thèmes une liste de documents choisis. Dans cette étape, le thème cible qui sert à étiqueter le document appartient à une liste restrictive de thèmes $T_i = \{T_{i_1}, T_{i_2}, \ldots, T_{i_{\mathbb{K}}}\}$. Les mots qui sont dans le document étiqueté sont tous affectés au thème cible $T_{i,j}$. Nous obtenons en sortie de cette étape un dictionnaire que l'on annotera $\mathbb{D}_t$ où pour chaque document étiqueté nous avons une liste de thèmes qui lui sont associés $\mathbb{D}_t[\Omega_i]$.

Pour les mots dans les autres documents non étiquetés, les thèmes qui leur seront associés seront échantillonnés de l'ensemble des thèmes K. Les deux niveaux de supervision de l'apprentissage donnent en sortie une liste restreinte de thèmes. Avec cette version semi-supervisée au moment du traitement d'un mot étiqueté ou d'un document étiqueté, nous appliquons l'échantillonnage de Gibbs uniquement sur cette liste restreinte de thème $T_{i,j}$ pour les mots et T_i pour les documents. L'échantillonage de Gibbs est appliqué pour approcher une nouvelle valeur de z_i avec l'équation 5.1. Cette equation représente la probabilité que le thème j est choisi pour un mot w_i étant donné tous les thèmes $-i$ choisi pour tous les mots dans le document d_i.

$$P(z_i = j \mid z_{-i}, w_i) \propto \frac{n_{-i,j}^{w_i} + \beta}{n_{-i,j}^{(\cdot)} + V\beta} \frac{n_{-i,j}^{d_i} + \alpha}{n_{-i,\cdot}^{d_i} + K\alpha} \tag{5.1}$$

Où $n_{-i,j}^{w_i}$ est le nombre d'occurrences du mot w_i dans le thème j. $n_{-i,j}^{(\cdot)}$ est le nombre d'occurrences de tous les autres mots dans le thème j. $n_{-i,j}^{d_i}$ est le nombre d'occurrences du thème j dans le document d_i. Enfin, $n_{-i,\cdot}^{d_i}$ est le nombre d'occurrences de tous les autres thèmes dans le document d_i.

Si nous souhaitons que l'échantillonnage de Gibbs soit uniquement appliquer sur la liste restreinte de thème $T_{i,j}$ pour les mots et T_i pour les documents, nous devons alors changer l'équation 5.1 et la remplacer par l'équation 5.2

$$P(z_i = j | z_{-i}, w) \propto \frac{n_{-i,j}^{w_i} + \beta}{n_{-i,j}^{(\cdot)} + \mathbb{V}\beta} \frac{n_{-i,j}^{d_i} + \alpha}{n_{-i,\cdot}^{d_i} + \mathbb{K}\alpha} \tag{5.2}$$

Où $\mathbb{K}$ et $\mathbb{V}$ sont calculés par les équations 5.3,5.3 au niveau des mots et par les équations 5.5 et 5.6 au niveau des documents. Notons que le comportement de l'échantillonnage de Gibbs sur des ensembles complets ou des sous-ensembles,

comme c'est le cas avec l'apprentissage semi-supervisé, est similaire [Koller 2009].

$$\mathbb{K} = |T_{i,j}| = len(\mathbb{W}_t[\omega_i]) \tag{5.3}$$
$$\mathbb{V} = |\Omega| \tag{5.4}$$

$$\mathbb{K} = |T_i| = len(D_t[\Omega_i]) \tag{5.5}$$
$$\mathbb{V} = V \tag{5.6}$$

Ces étapes sont résumé dans l'algorithme 10.

Algorithme 10 Version semi-supervisée de LDA

Précondition : w mots dans le corpus de document $D = (d_1, d_2, \ldots, d_n)$

Initialiser aléatoirement z
pour tout itérations **faire**
  pour tout $w \in d_i$, $d_i \in D$ et z_i dans z **faire**
    si $w \in \Omega$ **alors**
      Calculer $\mathbb{K}$ et $\mathbb{V}$ avec l'équation 5.3 et 5.4.
      pour tout thème j dans $\mathbb{K}$ thèmes **faire**
        Calculer $P(z_i = j|z_i, w)$ avec l'équation 5.2
      $z_i \sim p(z|.)$
    sinon si $d_i \in \mathbb{D}$ **alors**
      Calculer $\mathbb{K}$ et $\mathbb{V}$ avec l'équation 5.5 et 5.6.
      pour tout thème j dans $\mathbb{K}$ thèmes **faire**
        Calculer $P(z_i = j|z_i, w)$ avec l'équation 5.2
      $z_i \sim p(z|.)$
    sinon
      pour tout thème j dans $\mathbb{K}$ thèmes **faire**
        Calculer $P(z_i = j|z_i, w)$ avec l'équation 3.11
      $z_i \sim p(z|.)$
Calculer $\phi^{(z)}$
Calculer θ_d
retour de l'algorithme : $z, \phi^{(z)}, \theta_D$

Nous avons utilisé le résultat en sortie de la version semi-supervisée de LDA pour entraîner des forêts aléatoires, que nous présenterons dans ce qui suit.

5.2 Utilisation des forêts aléatoires

Les forêts aléatoires appartiennent à la catégorie des méthodes ensemblistes de l'apprentissage automatique [Rokach 2009]. L'appellation "forêts aléatoires" vient des travaux de [Ho 1995] qui a proposé un modèle d'agrégation d'arbres de décision qui sont construits sur des sous-ensembles aléatoires. Malgré sa simplicité d'implémentation, l'approche a rencontré des problèmes de surapprentissage pour lesquels

[Breiman 1996] a proposé une nouvelle approche appelée ensachage (*Bagging* qui est l'abréviation de *bootstrap agrégation* en anglais). La méthode consiste à prendre des échantillons avec remplacement, ensuite à entraîner un modèle d'apprentissage sur ces échantillons, et enfin à faire la moyenne de ces modèles. Ceci engendre, généralement, un modèle plus performant et plus résistant à la variation.

Les forêts aléatoires telles qu'elles sont connues maintenant ont été proposées par [Breiman 2001], et décrites comme un ensemble d'arbres de décisions entraînés. Elles sont une combinaison entre les arbres de décision et la méthode d'ensachage. Il est à noter que les forêts aléatoires améliorent la technique d'ensachage en réduisant la corrélation entre les arbres échantillonnés avec deux sources de stochasticité. La première vient de l'utilisation du bootstrap pour construire un ensemble aléatoire Z^* à partir des données d'entraînement. La deuxième source est la sélection aléatoire des attributs de séparation sur un noeud, où on tire un ensemble aléatoire avec remplacement N donné.

Algorithme 11 Construction d'une forêt aléatoire à partir de l'analyse sémantique semi-supervisée de LDA.

Précondition : Une collection de documents D avec C catégories.

1: **pour tout** arbre $b \in B$ arbres **faire**
2: Tirer un échantillon de bootstrap Z^* de taille C des données d'entraînement.
3: **tant que** la taille minimale c_{min} du noeud n'est pas atteinte **faire**
4: Tirer aléatoirement m variables des p variables.
5: Choisir le meilleur point pour diviser le noeud parmi les m variables.
6: Diviser le noeud en deux noeuds fils.
7: **retour** Un ensemble d'arbres $\{T_b\}_1^B$.

Nous décrivons dans l'algorithme 11 le processus de construction d'une forêt aléatoire à partir des résultats de l'analyse sémantique semi-supervisée de LDA. Le processus consiste à construire d'abord plusieurs arbres de décisions qui vont donner à chacun un résultat de prédiction de la catégorie $c \in C$ étant donné un document $d \in D$. Ensuite, pour éviter le sur apprentissage, nous effectuons deux interventions stochastiques : la première sur les données, et la deuxième sur le modèle.

L'intervention stochastique sur les données consiste à modifier légèrement l'échantillon des données d'entraînement pour chaque arbre de décisions. L'intervention stochastique sur le modèle consiste aux choix aléatoires des attributs lors de la division d'une branche dans un arbre de décision. Nous présentons dans la suite les données sur lesquels nous avons appliqué notre algorithme.

5.3 Application et analyse

Nous avons collecté le contenu textuel à partir des pages d'accueil de 60 sites web répertoriés dans les dossiers de *dmoz*[1]. Les répertoires web peuvent apporter un plus aux moteurs de recherche qui présentent, pour l'instant, les résultats selon un certain ordre de pertinence. Par exemple, une réponse à une requête de recherche qui inclue la hiérarchie des différentes catégories où se trouve la page web recherchée peut apporter à l'utilisateur une acuité par rapport à sa requête d'origine. Cependant, l'entretien de ces répertoires demande un effort humain considérable pour leur gestion. Notre approche peut contribuer à l'automatisation de cette tâche.

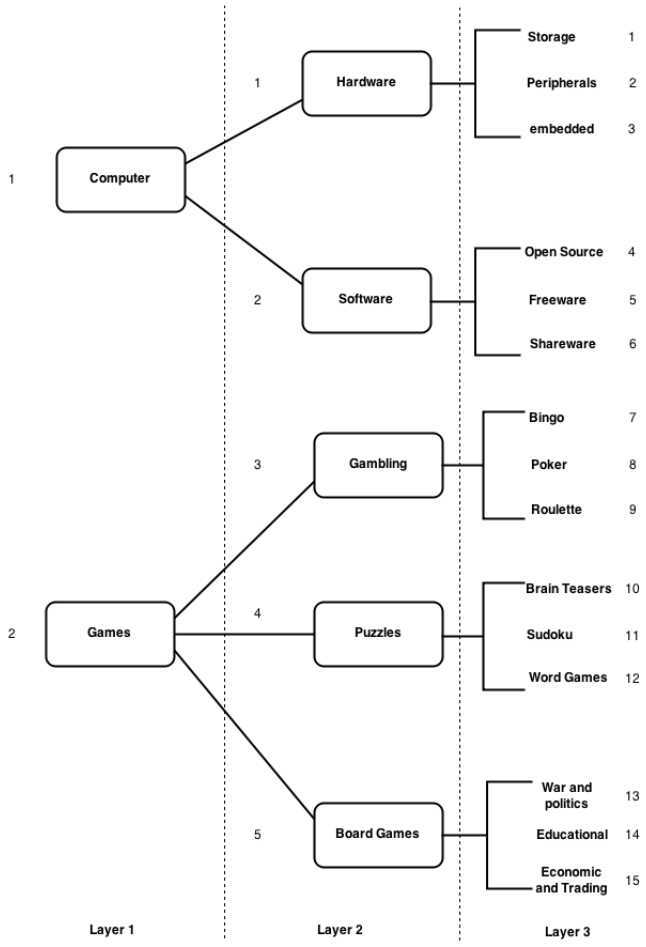


FIGURE 5.1 – Nous avons une structure avec 3 niveaux, 2 catégories principales, les catégories étant détaillées du niveau 1 au niveau 3.

Dans le dendrogramme présenté dans la figure 5.1, nous avons 15 catégories de bas niveau (niveau 3), 5 catégories (niveau 2) de mi-niveau, et 2 catégories de haut niveau. Pour chaque catégorie du niveau 3, nous avons 4 documents. Avant

1. www.dmoz.org

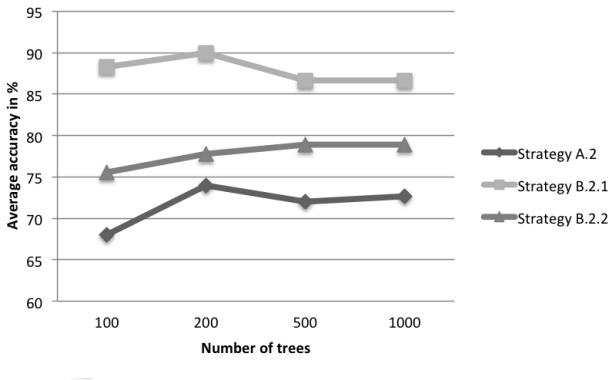


FIGURE 5.2 – La moyenne du taux de précision des stratégies de classification A et B.

d'exécuter la version semi-supervisée de LDA, nous avons nettoyé le texte de tous les mots vides, ponctuations, etc.

Après l'application de la version semi-supervisée de LDA nous obtenons un jeu de données où chaque page web est représentée par une distribution de probabilités qui est étiquetée avec les catégories prédéfinies. Nous avons séparé le jeu de données avec les proportions suivantes : 75% des données ont été dédiées pour l'entraînement des arbres de décision, cela afin de construire la forêt aléatoire. Les autres 25% des données ont été dédiées aux tests.

Dans notre évaluation, nous avons comparé deux stratégies de classification. La première est comparable à une classification basée sur les mots clés comme dans [Klassen 2010]. Au sein de cette stratégie, nous ne prenons pas en compte le résultat du niveau précédent. C'est-à-dire les prédictions pour la classification binaire du *Layer 1* dans la Figure 5.1. Nous avons dénoté cette stratégie par la lettre A.

La deuxième stratégie est ce que nous appelons une classification multi-niveaux. Avec cette stratégie nous prenons en compte les résultats de classification obtenus au niveau précédent. C'est-à-dire que les résultats sont pris comme entrée pour l'algorithme qui va classifier les documents dans le niveau suivant. Nous avons dénoté cette stratégie par la lettre B.

Pour la classification du premier niveau, nous avons obtenu un taux de classification égale à 93.93% avec un écart type nul. Cette classification a été exécutée à l'aide des caractéristiques obtenues avec la version semi-supervisée de LDA exécutée sur le niveau 3.

Nous avons également pris en compte dans notre analyse la variation du nombre d'arbres de décision dans la forêt aléatoire. Nous avons rapporté dans la table 5.1, la moyenne des résultats d'une dizaine d'exécutions de l'algorithme de classification basée sur la construction des forêts aléatoires.

Pour la stratégie de classification A, niveau 2, table 5.1, nous avons obtenu le taux de classification le plus bas 68% en moyenne pour un total de 100 arbres, et

	Niveau			
	100 arbres	**200 arbres**	**500 arbres**	**1000 arbres**
Min	93.3%	93.3%	93.3%	93.3%
Max	100%	100%	100%	93.3%
Moyenne	95.9%	95.9%	95.9%	93.3%
Ecart type	3.4	3.2	2.8	0
	Niveau 2 (stratégie A)			
	100 arbres	**200 arbres**	**500 arbres**	**1000 arbres**
Min	53.3%	66.6%	66.6%	66.6%
Max	80%	80%	73,3%	73,3%
Moyenne	68%	74%	71,9%	72,6%
Ecart type	6,8	5,8	2,8	2,1
	Niveau 2.1 (stratégie B)			
	100 arbres	**200 arbres**	**500 arbres**	**1000 arbres**
Min	50%	83.3%	83.3%	83.3%
Max	100%	100%	100%	100%
Moyenne	88.3%	89.9%	86.6%	86.6%
Ecart type	15.8	8.6	7	7
	Niveau 2.2 (stratégie B)			
	100 arbres	**200 arbres**	**500 arbres**	**1000 arbres**
Min	66.6%	66.6%	66.6%	77.7%
Max	88.8 %	100%	88.8%	88.8%
Moyenne	75.5%	77,7%	78,8%	78,8%
Ecart type	7	10.4	8.1	3.5

TABLE 5.1 – Investigation des paramètres des forêts aléatoires sur le taux de classification hiérarchique selon plusieurs stratégies.

pour le taux de classification le plus haut 74% en moyenne pour un total de 200 arbres.

Pour la stratégie B, où nous avons pris en compte le résultat de la classification au niveau de le niveau 1, nous pouvons diviser la tâche de classification et donner en entrée des caractéristiques qui correspondent à la division dans le niveau 1. Par exemple, si nous savons qu'un document est classifié dans la catégorie *computer*, l'algorithme de classification ne recevra que les caractéristiques liées à cette catégorie.

Dans la table 5.1, nous avons deux parties. Le niveau 2.1 qui correspond aux sous-catégories de la catégorie *computer*, et le niveau 2.2 qui correspond aux sous catégories de la catégorie *Games*, tous deux illustrées dans la figure 5.1. Pour la stratégie de classification B, niveau 2.1, table 5.1, nous avons obtenu un minimum de 86% en moyenne à partir de 500 arbres, et un maximum de 89% en moyenne pour 200 arbres.

Pour la stratégie de classification B, niveau 2.2, table 5.1, nous avons obtenu un minimum de 75% en moyenne pour 100 arbres et un maximum de 78.8% en moyenne à partir de 500 arbres. En conclusion et selon les résultats obtenus, plus il y a de thèmes ou de catégories à classifier, moins l'algorithme de classification basé sur les forêts aléatoires est précis. Cette conclusion est aussi mentionnée dans les travaux de [Klassen 2010].

Dans cette partie, nous avons proposé une méthode pour la classification d'un document selon une hiérarchie de classe représentée en dendrogramme. Les contri-

butions majeures reliées à cette méthode sont :

- l'extraction des caractéristiques sémantiques avec une approche semi-supervisée.
- La sélection des caractéristiques avec la construction de forêt aléatoire.
- La proposition d'une méthode pour une classification à la fois «souple» et multi-niveau.

Observons que les expérimentations ont été effectuées sur une collection de documents D réduite, ceci à cause du problème du passage à l'échelle lors de l'utilisation de l'allocation latente de Dirichlet. Dans la prochaine section, nous proposerons un outil pour offrir des solutions parallèles.

La deuxième méthode nous a permis de classifier à document à travers une hiérarchie de document en partant de la catégorie la plus générale vers ceux qui sont particuliers au document en question. Au sein de cette méthode, nous avons proposons une version semi-supervisée de l'allocation latente de Dirichlet que nous avons combinée avec les forêts aléatoires. Nous avons été capables de classifier les documents avec une moyenne de 95.9% pour les catégories les plus générales et une moyenne de 89.9 % pour les catégories les plus particulières.

Chapitre 6

Construction de corpus annotés pour le dialecte tunisien

Résumé : Avec l'augmentation constante du nombre d'utilisateurs arabophones sur les réseaux sociaux tels que Twitter et Facebook, il s'est créé parallèlement un besoin pour des systèmes automatiques de traitement de la langue arabe, par exemple, les systèmes d'analyse de sentiments. Ces derniers sont utilisés pour classifier les textes dans des catégories qui correspondent à un sentiment : joyeux, triste, neutre. Ou bien encore, dans les systèmes de traduction de l'arabe à d'autres langues. Cependant, et à cause de la diversité des dialectes dans la langue arabe, il y a un manque de données d'entraînement pour la construction de ce type de systèmes. Un texte écrit en Tunisien ne peut pas être directement traduit en anglais si le système de traduction est entraîné avec de l'arabe moderne, de même pour l'analyse de sentiment. Une première étape consiste alors à différencier l'arabe moderne des dialectes qui existent, une seconde étape consiste à entraîner pour chaque dialecte un système différent. Nous proposons, dans ce chapitre, une étude du dialecte tunisien en fournissant un corpus de données pour les applications de différenciation et d'analyse des sentiments.

Ces dernières années, nous avons constaté un intérêt croissant, dans la communauté du traitement de la langue arabe, pour la fouille d'opinion et l'analyse des sentiments [Abdul-Mageed 2011, Shaalan 2014, Salameh 2015, Mohammad 2015]. Cependant, cette tâche reste difficile à réaliser à cause de la diversité de la langue arabe et de la multitude de dialectes.

Un outil qui est construit pour un dialecte ne marchera pas forcément pour un autre. Par exemple, l'outil de segmentation proposé par [Monroe 2014], ou l'outil d'analyse morphologique proposé par [Pasha 2014], sont deux outils construits sur l'arabe standard moderne (ASM) et le dialecte égyptien (EGY). L'application de ces outils sur le dialecte tunisien (TN), par exemple, ne donne pas des résultats précis. Car le TN a sa propre structure. Pour une analyse plus détaillée des différentes morphologies des dialectes arabes, se référer aux travaux de [Malmasi 2015]. Il y a donc un besoin de collecter plus de données de différents dialectes afin de construire un outil générique.

La contribution de cette partie se résume dans les trois points suivants :

1. Nous proposons un jeu de données annoté manuellement pour la tâche de l'analyse des sentiments (e.g. positifs, négatifs, neutres). Il est composé d'un ensemble de phrases en dialecte tunisien (TN) et en arabe standard moderne (ASM).

2. Nous avons comparé la performance de six algorithmes de classification avec l'utilisation du gain d'informations comme méthode pour la sélection des caractéristiques. Les résultats de cette partie peuvent être considérés comme un référentiel (parangonnage) pour d'autres travaux qui utiliseront notre jeu de données.
3. Nous avons analysé la différence entre les résultats de classification des algorithmes entraînés seulement sur le TN et ceux entraînés sur un mélange entre le TN et le ASM.

Travaux	Nom	Taille	ASM/Dialecte	Source
[Aly 2013]	LSABR	63,257	MSA/Non mentionné	GoodReads.com
[ElSahar 2015]	HTL	15,572	MSA/Non mentionné	TripAdvisor.com
[ElSahar 2015]	RES	10,970	MSA/Non mentionné	Qaym.com
[Nabil 2015]	ASTD	10,000	MSA+égyptien	Twitter.com
[Refaee 2014]	ATC	8,868	MSA+JO	Twitter.com
[ElSahar 2015]	PROD	4,272	MSA/non mentionné	Souq.com
[Abdul-Mageed 2014b]	MONTADA	3,097	MSA/LEV/EGY	Forums
[Abdul-Mageed 2014b]	TGRD	3,015	MSA/non précisé	Twitter.com
[Abdul-Mageed 2014a]	THRIR	3,008	MSA/LEV/EGY	Wikipedia TalkPages
[ElSahar 2015]	MOV	1,524	MSA/non mentionné	Elcinemas.com

TABLE 6.1 – Jeux de données existants ordonnés par leur taille.

Dans la table 6.1, nous citons la liste des jeux de données collectées pour la tâche de l'analyse des sentiments. Trois de ces jeux de données [Nabil 2015, Refaee 2014, Abdul-Mageed 2014b] sont collectés du site de microblogging Twitter et par la suite annotés. Nous trouvons, dans le travail de [Nabil 2015], un référentiel construit avec quatre algorithmes de classification, et aucune distinction entre le ASM et le EGY. Dans le travail de [Refaee 2014], Les auteurs proposent un jeu de données constitué de tweets jordaniens, ils détaillent dans leur article la méthodologie d'annotation utilisée ainsi que des statistiques reliées à leur jeu de données. Enfin, dans le travail de [Abdul-Mageed 2014b], les auteurs étudient le lexique des tweets qu'ils ont collectés et présentent une analyse des sentiments subjectifs dans la langue arabe.

6.1 Jeu de données proposées et statistiques

Traduction en français	Arabe
Elections présidentielles	الانتخابات الرئاسية
Elections législatives	الانتخابات التشريعية
Elections tunisiennes	الانتخابات التونسية
Deux des principaux *hashtags* utilisés par la communauté tunisienne sur Twitter pour parler des élections	#tnelec
	#tnprez

TABLE 6.2 – Exemples des mots clés/*hashtags* utilisés lors de la collecte des données sur Twitter.

	Label	Definition	Example	English Translation
TN	**Positif**	Avec un indicateur positif utilisé	برَافُو نورَا.	Bravo ! Noura.
	Négatif	Avec un indicateur négatif utilisé	مَاينجمش يحكُم وحدو و لو حب.	Il ne peut pas gouverner seul même s'il le voulait
	Neutre	Pas d'indicateurs d'émotions utilisées	كيفَاش التوَانسة المقيمين بَالخَارج ينتَخبو	Comment les tunisiens expatriés votent-ils ?
MSA	**Positif**	//	بعثة تصف الَانتخَابَات التونسية بأَنهَا تعدُدِية و شفَافة.	Une représentation officielle décrit les élections comme étant transparentes et diverses.
	Négatif	//	مرَاقبو الريَاسية يتذمرون من عدم وصول بطقَاتهم	Les observateurs des élections présidentielles se plaignent de ne pas avoir reçu leurs cartes.
	Neutre	//	التفَاصيل في مقَال الجزيرة	Les détails sont dans l'article de Aljazira.

TABLE 6.3 – Exemple des tweets annotés.

À l'aide du service logiciel *Twitter Streaming API*, fourni par Twitter, nous avons collecté un total de 50,000 tweets. Ces tweets ont été publiés de façon publique entre la période du 1^{er} octobre 2014 et le 23 décembre 2014. La première date correspond aux élections des 217 sièges à l'Assemblée nationale, la deuxième date est postérieure aux élections présidentielles.

Après un traitement des tweets et la suppression des tweets écrits avec des lettres en Latin, nous avons obtenu un total de 10,00 tweets écrits entièrement en lettres arabes. La constitution du jeu de données est fondée sur la collecte de certains mots clés qui ont été testés manuellement sur la barre de recherche du site de mircoblogging Twitter, et ont été discutés avec différents membres de la communauté tunisienne. En plus des mots clés, nous avons ajouté les *hashtags* comme une entrée lors de la requête des tweets. Les *hashtags* sont un moyen de donner plus de visibilité à un tweet. Ils sont considérés comme un hub où tous les tweets qui traitent du même sujet peuvent être facilement identifiés. Nous avons représenté les mots clés utilisés ainsi que les hashtags dans la table 6.2.

Nous avons, avec trois personnes parlant le Tunisien, annoté manuellement 5514 exemples en ne tenant compte que des indicateurs positifs ou négatifs. Si un de ces indicateurs n'est pas identifié dans le tweet, nous le considérons comme neutre (se référer à la table 6.3 pour des exemples). Dans le cas où les annotateurs sont incapables de se décider sur un tweet, ils le suppriment tout simplement. Nous n'avons pas, de ce fait, considéré les sens cachés ou les phrases sarcastiques. Tous les tweets qui contenaient une redondance ou de la publicité ont été aussi supprimés.

Le jeu de données proposé contient donc 5514 tweets annotés manuellement. 3760 de ces tweets sont en arabe standard moderne (ASM) et 1754 en dialecte

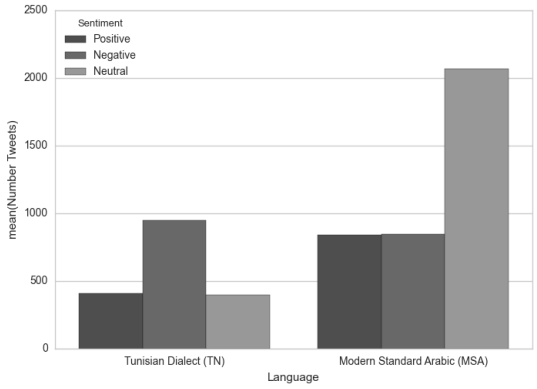


FIGURE 6.1 – Nombre de tweets pour chaque catégorie.

Nb. Tweets	5514
Nb. mots	49940
Nb. max mots / Tweet	27
Nb. min mots / Tweet	1
Moy. mots / Tweet	10
Taille du vocabulaire	10553

TABLE 6.4 – Statistiques du contenu du jeu de données annoté.

tunisien (TN). La proportion des différentes classes est illustrée dans la figure 6.1. Pour les tweets en ASM, nous avons un nombre important de tweets neutres, cela s'explique par le fait que c'est une langue utilisée pour les journaux écrits et télévisés. Pour les tweets en TN, nous avons un nombre important de tweets négatifs. Les tweets positifs et négatifs sont presque similaires en nombre. Une analyse poussée, des raisons expliquant ces différentes proportions, est au-delà de la portée de nos travaux. Nous résumons, dans la table 6.4, quelques statistiques relatives à notre jeu de données.

6.2 Sélection de caractéristiques et algorithmes de classification

MSA/TN			**Nb. Grammes**	**Nb. Grammes GI**	**Ratio**
MSA+TN	$GI > 0.0$	1g	10553	406	3.85%
		1g+2g	33748	724	2.14%
		1g+2g+3g	58110	940	1.62%
TN	$GI > 0.0$	1g	7372	46	0.62%
		1g+2g	21803	56	0.25%
		1g+2g+3g	35862	57	0.15%

TABLE 6.5 – Sélection des caractéristiques sur différents n-grammes avec le ratio de sélection.

Pour l'extraction et la sélection des caractéristiques, nous avons utilisé le gain d'informations (GI). GI est une méthode qui est utilisée pour extraire les caractéristiques les plus importantes par rapport aux attributs de la classe que l'on cherche à trouver. GI calcule la décroissance attendue de l'entropie ou de l'incertitude face à l'arrivée de nouvelles informations. Quand l'entropie décroît, l'information croît. GI est décrite par l'équation suivante :

$$Info(S) = -\sum_{i=1} m(P_i) log_2(P_i) \tag{6.1}$$

où S est l'ensemble des instances, P est la probabilité qu'une instance choisie aléatoirement appartienne à une classe i. Enfin, m est le nombre de classes (e.g. nous avons 3 classes : positif, négatif, neutre).

Pour calculer le GI, nous avons besoin de mesurer le nombre de bits nécessaires pour encoder l'information relative à la classification d'une instance dans S, en utilisant une caractéristique F. Cette quantité est mesurée par :

$$Info_F(S) = -\sum_{j=1}^{v} \frac{|S_j|}{S} \times Info(S_j) \tag{6.2}$$

$$IG(F) = Info(S) - Info_F(S) \tag{6.3}$$

Où v est le nombre de partitions, $\frac{|S_j|}{S}$ est le poids de la partition (classes) j et $Info(S_j)$ est l'entropie de S_j calculée avec l'équation 6.2.

Dans ce travail, nous avons choisi les caractéristiques avec une valeur au-dessus de 0.0, relative au GI. Dans la table 6.5, nous comparons deux types d'extraction de caractéristiques. Le premier est appliqué sur les 5514 tweets composant notre jeu de données. Nous avons obtenu 10553 grammes uniques, séparés par des espaces dans chaque tweet.

Après l'extraction des caractéristiques, nous avons sélectionné 406 grammes uniques avec un GI au-dessus de 0.0, comme "نحبها" en dialecte tunisien que nous traduisons en français par "Je l'aime". Nous avons répété ce processus pour les caractéristiques avec un et deux grammes, par exemple, "اتعس من" qui veut dire "Pire que". La deuxième partie de la table 6.5 est l'application de l'extraction et de la sélection des caractéristiques sur le dialecte tunisien. Le ratio dans la table 6.5 représente l'agressivité de la sélection des caractéristiques. Plus la valeur est petite, plus la sélection de caractéristiques est agressive.

Dans la suite, nous présentons brièvement les algorithmes que nous avons utilisés pour construire notre référentiel de comparaison sur le jeu de données que l'on propose.

Classification naïve bayésienne : la classification naïve bayésienne consiste en l'application de la règle de Bayes sur les documents (e.g. de type tweets) et les classes (e.g. négatifs ou positifs). Pour un document d et une classe c, la règle de Bayes s'écrit comme suit :

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \tag{6.4}$$

$$P(c, x_1, ..., x_n) = P(c)P(x_1|c)...P(x_n|c) \tag{6.5}$$

L'équation 6.5 représente l'indépendance conditionnelle où c est une classe binaire, par exemple positive ou négative. x est une caractéristique, par exemple un mot dans un tweet. La probabilité qui correspond à l'ensemble des caractéristiques est indépendante étant donnée la classe c.

Séparateurs à Vaste Marge (SVM) SMV ou encore machines à point de support sont une méthode de classification supervisée binaire. Les SVM ont été introduits par Vapnik dans son ouvrage sur la théorie de l'apprentissage statistique. Conçue comme un classifieur linéaire, la méthode consiste à chercher un hyperplan qui sépare deux classes de données de manière à maximiser la marge entre les exemples les plus proches de chacune des deux différentes classes ; ces exemples sont appelés vecteurs support.

Méthodes des k-plus proches voisins (NN) NN utilise le vote en majorité pour associer un objet à la classe de ses k plus proches voisins. Dans nos expérimentations nous avons utilisé $k = 1$.

Arbre de décisions. Un arbre de décision est un graphe qui possède une seule racine et un ensemble de feuilles. En entrée, l'arbre reçoit une description de la situation avec un ensemble de propriétés et en sortie il donne une décision de 0 ou 1. Nous avons utilisé dans nos expérimentations deux méthodes ; proposées par Weka 3.6 [1]. La première est appelée **J48**, c'est une implémentation en Java de l'algorithme C4.5 [Salzberg 1994]. La deuxième est appelée **PART**, construit partiellement un arbre de décision de type C4.5 et génère une liste de décisions.

Forêts aléatoires (RF) les forêts aléatoires peuvent être considérées comme un ensemble d'arbres de décisions entraînés aléatoirement et combinés avec la méthode d'ensachage qui réduit le surapprentissage. Pour plus de détails, veuillez vous référer à la section 5.2.

6.3 Résultats

Afin de fournir un référentiel pour l'utilisation du jeu de données que nous avons proposé, nous avons d'abord appliqué les algorithmes de classification sur les caractéristiques sélectionnées à partir d'un mélange de tweets TN et ASM. Ensuite, nous avons entraîné les algorithmes de classification avec des caractéristiques sélectionnées uniquement à partir de tweets TN.

Pour les algorithmes entraînés avec des caractéristiques TN, nous avons exploré les résultats de sortie sur trois classes (positives, négatives, neutres) mais aussi sur deux classes (positives, négatives). Les résultats des expérimentations sont rapportés dans la table 6.6. La fiabilité des algorithmes cités dans la table a été estimée avec

1. weka.sourceforce.net

MSA+TN/ 5514 tweets		**Precision moy.**	**Rappel moy.**	**F1 moy.**	**Justesse**
NB	1g	0.50	0.50	0.50	49.34%
SVM	**1g+2g+3g**	**0.54**	**0.53**	**0.53**	**53.55%**
NN	1g+2g	0.50	0.50	0.49	50.11%
RF	1g+2g	0.51	0.51	0.51	51.81%
J48	1g+2g	0.49	0.49	0.47	49.58%
PART	1g	0.50	0.50	0.48	50.08%
TN/ 1754 tweets		**Precision moy.**	**Rappel moy.**	**F1 moy.**	**Justesse**
NB 3c	1g	0.49	0.53	0.45	53.42%
NB 2c	1g	0.65	0.69	0.64	69.61%
SVM 3c	**1g**	**0.47**	**0.53**	**0.41**	**53.47%**
SVM 2c	**1g+2g**	**0.67**	**0.71**	**0.63**	**71.09%**
NN 3c	1g	0.47	0.53	0.42	53.24%
NN 2c	1g+2g	0.68	0.70	0.61	70.72%
RF 3c	1g	0.47	0.53	0.43	53.42%
RF 2c	1g+2g+3g	0.67	0.70	0.63	70.87%
J48 3c	1g	0.47	0.53	0.40	53.64%
J48 2c	1g	0.66	0.70	0.62	70.42%
PART 3c	1g+2g	0.40	0.54	0.41	54.04%
PART 2c	1g	0.67	0.71	0.63	71.01%

TABLE 6.6 – Les résultats des classifieurs entraînés avec des caractéristiques du dialecte Tunisien. L'abréviation g est pour -gramme (le modèle de langage) et l'abréviation c pour classes.

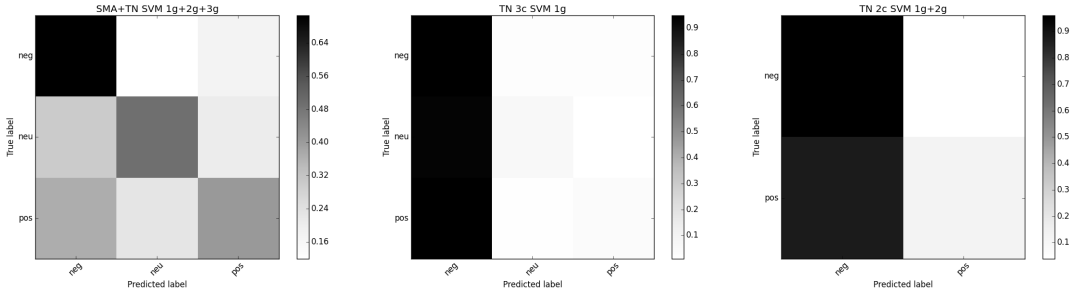
la méthode de la validation croisée sur 10 échantillons. Les meilleurs résultats ont été obtenus avec un SVM qui a été entraîné sur un ensemble composé d'un gramme, deux grammes et trois grammes.

Les dialectes arabes et spécifiquement le dialecte tunisien contiennent plusieurs mots provenant de l'ASM. Nous voulions dans nos expérimentations voir si le fait d'entraîner les algorithmes uniquement sur le TN aurait un effet sur la justesse de la discrimination de la classe d'un tweet. En observant les résultats dans la table 6.6, nous constatons une augmentation de la justesse des résultats de tous les algorithmes entraînés avec des caractéristiques sélectionnées uniquement à partir du dialecte TN.

Nous rapportons également, dans la table 6.6, le résultat de la discrimination des tweets en deux classes (positive et négative). La réduction du nombre de classes fait augmenter naturellement la justesse. Les meilleurs résultats sont encore obtenus avec le SVM pour les trois classes (3c) avec un gramme en caractéristique. Pour les deux classes (2c), les meilleurs résultats sont obtenus avec un SVM entraîné avec un ensemble de caractéristiques constitué de un et deux grammes.

6.4 Discussion de l'effet du nombre de caractères

L'algorithme de classification SVM a prouvé son efficacité à travers une validation croisée sur 10 échantillons. Pour analyser ces résultats, nous avons illustré dans la figure 6.2 trois matrices de confusion pour comparer la justesse des résultats donnée dans la table 6.6 avec les résultats de faux négatifs et de faux positifs donnés par la matrice.



(a) (b) (c)

FIGURE 6.2 – (a) Matrice de confusion pour un SVM entraîné avec 3 grammes extraits d'un mélange du dialecte tunisien et de l'arabe moderne. (b) Matrice de confusion pour un SVM entraîné avec 1 gramme extrait du dialecte tunisien. (c) Matrice de confusion pour un SVM entraîné avec 1 gramme extrait du dialecte tunisien.

La matrice illustrée dans la figure (a) 6.2, représente les exemples prédits par un SVM qui a été entraîné sur des caractéristiques provenant de ASM et du dialecte TN. Dans cette matrice, nous avons une importante proportion de vrais négatifs et une bonne proportion de vrais neutres. Les tweets classés positivement ont presque la même proportion de vrais positifs et de faux positifs.

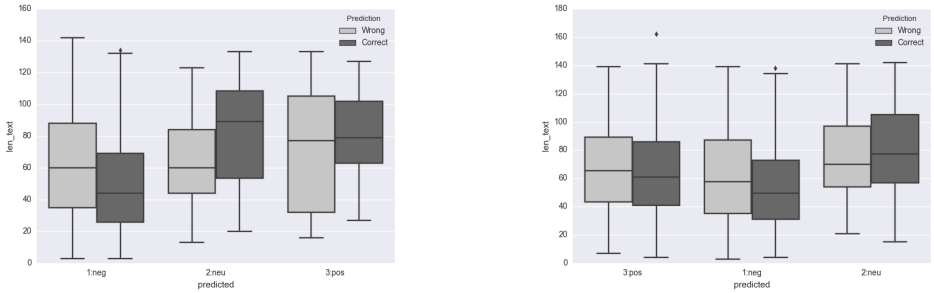
Les matrices de confusion (b) et (c) de la figure 6.2 représentent les exemples prédits par un SVM entraînés uniquement sur le TN. (b) est la matrice de confusion avec trois classes (positif, négatif et neutre). (c) est la matrice de confusion avec deux classes (positif et négatif).

Pour les deux matrices, nous pouvons clairement identifier une grande proportion de faux positifs pour la classe positive, et une grande proportion de vrais négatifs pour la classe négative. Cela veut dire qu'on une tendance de justesse pour la classe négative et aussi de meilleures performances de classification pour cette dernière, sachant aussi que nous avons un grand nombre de tweets négatifs donnés lors de l'entraînement.

Nous avons également proposé une analyse de la relation entre la longueur des tweets et les meilleures prédictions de classification que nous avons obtenues avec SVM. Dans la figure 6.3, nous avons illustré différentes boîtes à moustache, ceux en gris foncé représentent les prédictions correctes du classifieur et ceux en gris clair représentent les prédictions erronées.

La figure (a) 6.3 représente les prédictions des trois classes données par un SVM entraîné sur le dialecte tunisien. La figure (b) 6.3 représente les prédictions des trois classes données par un SVM entraîné sur le dialecte tunisien et l'arabe standard moderne.

Dans la figure(a) 6.3, 50% des prédictions correctes pour la classe positive ont été effectuées sur des tweets qui ont une longueur qui se situe entre 10 et 60 caractères. Les prédictions erronées ont été effectuées sur des tweets d'une longueur inférieure



(*a*) (*b*)

FIGURE 6.3 – Relation entre les résultats de prédiction lors de la classification et la taille des tweets. (a) SVM entraîné sur les tweets TN. (b) SVM entraîné sur les tweets ASM.

à 80 caractères. Cette différence dans les prédictions selon la longueur nous ouvre plusieurs possibilités d'améliorations d'un modèle d'apprentissage.

Dans la figure (b) 6.3, nous avons noté que les boîtes à moustaches correspondent aux prédictions correctes et celles qui sont erronées sont presque similaires. La longueur médiane des tweets pour la classe positive est aux alentours de 65 caractères. Nous pouvons conclure que pour les classes entraînées avec un mélange de SMA et du dialecte TN les résultats de prédictions sont presque meilleurs que des prédictions faites aléatoirement.

Dans cette partie, nous avons proposé un jeu de données collectées à partir de Twitter pour la tâche de détection de sentiments. Nous avons d'abord annoté manuellement les tweets avec trois classes et séparé les tweets en arabe standard moderne et ceux écrits en dialecte tunisien.

Ensuite, nous avons construit un référentiel de comparaison avec six algorithmes de classification et présenté notre méthodologie pour la sélection des caractéristiques. Ce référentiel pourra servir à d'autres chercheurs pour de futures comparaisons.

Suite à l'analyse des résultats que nous avons faite, nous avons quelques idées pour améliorer les résultats de classification. Dans nos futurs travaux sur cette partie, nous projetons d'étendre le jeu de données et de proposer un entraînement des algorithmes fondé sur le lexique du dialecte tunisien.

Dans cette partie, nous avons traité la problématique de la classification et de la discrimination des documents en tenant compte des thèmes qu'ils relatent. Nous avons approché le problème d'abord en construisant un espace sémantique latent en utilisant la méthode de l'allocation latente de Dirichlet puis, en exploitant la notion de "voisinage" en proposant des mesures de proximité entres les entités dans ces espaces.

Nous avons proposé deux méthodes de classification. La première permet de regrouper les documents selon deux critères : le premier est qualitatif ; il est représenté par les thèmes majeurs dans la collection des documents. Le deuxième est quantitatif ; il est représenté par la distance entre la distribution des thèmes dans chaque document de la collection. Au sein de cette méthode, nous avons procédé à l'analyse de la structure de l'espace sémantique latent à l'aide des axiomes de la prétopologie. Cela nous a permis de fournir, avec la première méthode de classification, le nombre de classes ou de groupes ainsi que leurs centroïdes respectifs. Cela nous évite une initialisation aléatoire de ces paramètres et nous permet d'obtenir une meilleure qualité dans les regroupements de documents.

La deuxième méthode nous a permis de classifier les documents à travers une hiérarchie (de catégories) en partant de la catégorie la plus générale vers celles qui sont particulieres au document en question. Au sein de cette méthode, nous avons proposé une version semi-supervisée de l'allocation latente de Dirichlet que nous avons combinée avec les forêts aléatoires. Nous avons été capables de classifier les documents avec une moyenne de 95.9% pour les catégories les plus générales et une moyenne de 89.9 % pour les catégories les plus particulières.

Nous avons également proposé dans cette partie une version distribuée de l'allocation latente de Dirichlet avec une parallélisation de l'algorithme de l'échantillonnage de Gibbs. Cela afin d'assurer la bonne gestion du passage à l'échelle des deux méthodes que nous avons proposées ultérieurement.

À la fin de cette partie, nous avons proposé une vérité terrain pour la classification et la distinction du dialecte tunisien dans les textes arabes et avons levé les problématiques liées au traitement d'une langue non latine. Dans la partie suivante de la thèse, nous nous intéresserons aux textes numérisés, c'est-à-dire aux images des textes où nous allons étudier la tâche de l'identification des scripts.

Deuxième partie

Analyse automatique des textes manuscrits

Chapitre 7

Identification des scripts dans les manuscrits anciens

Résumé : La reconnaissance de scripts est la distinction d'une langue écrite par rapport à une autre dans un manuscrit ancien. Les manuscrits anciens sont des documents complexes de par la diversité de leurs contenus, par exemple la présence de lettrines, artefacts et différentes images. Mais aussi, par le niveau de détérioration qu'ils ont subit qu'elle soit au travers d'interventions humaines ou naturelles. Cette complexité rend la tâche de l'identification des scripts difficile, car l'algorithme doit avoir une bonne capacité de généralisation. Par exemple, l'algorithme doit être capable de reconnaître le script d'un ou plusieurs fragments provenant du même manuscrits, mais éparpillés dans plusieurs sources de données. Notre approche consiste à aborder le problème de l'identification des scripts par la détermination des caractéristiques en prenant le signal de l'image du document dans sa globalité, prenant donc en compte l'environnement où se trouve le script. Nous faisons l'hypothèse qu'un apprentissage effectif des caractéristiques utiles à la classification est possible dans un premier temps. Et, dans un second temps, l'extraction de caractéristiques fines est abordable par un empilement de réseaux de neurones de types auto-encodeurs (CAE pour Convolutional Auto-Encoders) afin de fournir une représentation alternative des données reçues en entrée. Ces représentations fines mélangent les propriétés du script aux propriétés de la page où ils se trouvent. Les représentations sont des plus générales dans les premières couches, par exemple le changement de contraste dans l'image, vers les plus particulières, par exemple le tracé d'un scribe (écrivain) par rapport à un autre. Les résultats obtenus avec les représentations CAE sont comparables à ceux fabriqués par des experts. Nous avons rapporté les résultats obtenus avec les caractéristiques apprises, et ce en les comparants aux caractéristiques extraites à l'aide d'une expertise humaine. Nous avons enfin relevé l'importance de l'initialisation des paramètres lors de l'apprentissage.

7.1 Introduction

7.1.1 Position du problème

L'évolution technologique de l'acquisition d'images, par scanner ou appareil photo de type réflexe, a permis de réduire les coûts liés à la numérisation de nombreux documents anciens, dans les bibliothèques, archives municipales, musées, etc. Ceci a résulté en une rapide expansion du nombre de travaux sur l'analyse d'images de documents ou DIA, *Document Image Analysis*. Le DIA est un domaine qui se trouve au croisement de l'analyse d'images, la reconnaissance de formes et les

(*a*) (*b*)

(*c*) (*d*)

FIGURE 7.1 – Exemple des quatre scripts correspondant aux quatre classes identifiées (a) Éthiopien (b) Syriaque (c) Grec (d) Latin

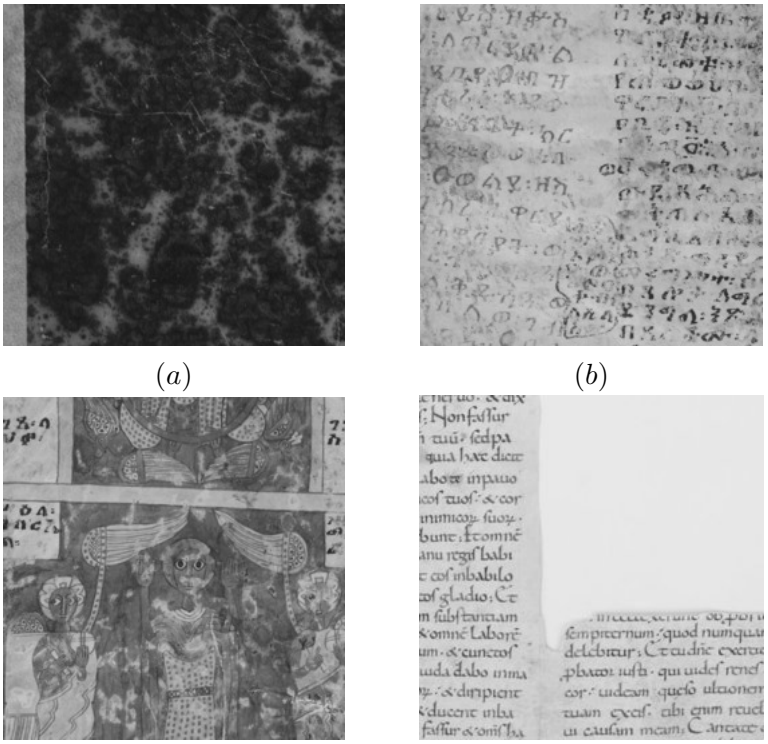
sciences humaines, avec par exemple l'implication de la philologie ou encore la paléographie.

Dans ce chapitre, nous nous intéressons à un cas particulier de DIA qui est l'identification des *scripts*[1] Dans les pages d'un manuscrit ancien. Le but d'exécuter une telle tâche est de proposer une méthodologie qui permettra de rechercher un document en émettant une requête sur son *script*, ou encore de transcrire son contenu avec un algorithme adapté de reconnaissance de caractères OCR *Optical Character Recognition* en anglais.

La figure 7.1 montre quatre scripts issus de différents manuscrits anciens qui seront étudiés dans ce chapitre (voir la table 7.1). Nous appelons donc script, un texte écrit à la main, à ne pas confondre avec une langue qui est la lecture du texte écrit ou le parlé dans une discussion.

Les manuscrits anciens sont des documents complexes. D'abord, leur mise en forme est complexe avec une variété d'artefacts comme des lettrines, des dessins, et des contenus textuels de différentes formes (e.g. figure 7.2 a et c). Ensuite, leur qualité est souvent détériorée par le temps, ou par des interventions humaines (e.g taches, coupures) comme cela est illustré par exemple dans les figures 7.2 (b) et 7.2 (d)). Puis, vu qu'il n'existe pas encore de normes communes à l'ensemble des

1. À ne pas confondre avec le terme "script" d'un langage informatique.



(a) (b) (c) (d)

FIGURE 7.2 – Exemple de différentes formes de dégradation, artefacts, intervention humaine que l'on peut trouver dans les manuscrits étudiés. (a) Une page de garde noire. (b) Taches sur la page et dilution de l'encre. (c) Artefacts. (d) Coupures et interventions humaines.

bibliothèques (e.g. Cambridge, Stanford, Bibliothèque National de France, etc.), la numérisation des manuscrits qui sont complets est souvent non homogène, c'est-à-dire que parfois le code couleur lors de la numérisation n'est pas le même, ou que la résolution n'est pas la même. Enfin, nous pouvons également avoir affaire à plusieurs fragments d'un seul manuscrit qui sont distribués dans plusieurs bibliothèques à travers le monde.

En pratique, les scripts sont qualifiés à l'aide d'un ensemble de caractéristiques (e.g. largeur du trait, contraste, forme des lettres). Ces caractéristiques sont accessibles au travers du traitement de l'image représentée par une matrice de pixels. Le traitement aux termes des pixels s'effectue par l'application d'un filtre sur un ou plusieurs niveaux qui correspondent à différentes régions d'intérêts (ROI pour *Region Of Interest* en anglais). Ces ROI peuvent représenter une ligne, un mot, ou encore un caractère. Dans ce cas, les documents illustrés dans la figure 7.2 sont le plus souvent considérés comme du bruit et sont supprimés lors de l'extraction des caractéristiques, et aussi de la discrimination des scripts.

En raison de la nature complexe des documents anciens, nous pensons que les méthodes qui sont entraînées sur un seul manuscrit avec des caractéristiques ex-

traites au niveau des ROI cités perdent en information et ne sont pas directement applicables à d'autres manuscrits anciens. Dans cette partie, nous proposons de travailler au niveau de la page sans aucun nettoyage ni pré traitement préalable sur les pixels du manuscrit.

7.1.2 Spécification de notre jeu de données

Dataset	#Pages	Scripts	Code manuscrit
BNF_eth_226	222	Ethiopic	Eth 226
BNF_eth_248	188	Ethiopic	Eth 248
BNF_eth_31	169	**Ethiopic**	Eth 31
BNF_eth_32	415	**Ethiopic**	Eth 32
BNF_syr_438	217	**Syriaque**	Syr 438
N41_BNF_grec_2465	476	**Greek**	Gr 2465
N97jpg	93	Greek	Suppl Gr 687
N43jpg	529	Greek	Gr 2795
N65jpg	377	**Greek**	N/A
N71jpg	417	Greek	Gr 164
N73jpg	824	Greek	N/A
N84	531	**Latin**	Latin 2246
N89jpg	338	**Latin**	Latin 9452

TABLE 7.1 – Description du jeu de données utilisé. Les scripts qui sont en gras sont ceux qui ont été utilisés lors de la phase d'entraînement pour l'identification des scripts.

Pour mener notre étude sur l'identification des scripts, nous avons choisi d'utiliser un jeu de données composé de 13 manuscrits avec un total de 4796 pages divisées en quatre principaux scripts : Éthiopien, Syriaque, Grec et Latin. Des exemples représentés par des captures sont illustrés dans la figure 7.1. Les manuscrits ont été téléchargés sur le site de la Bibliothèque Nationale de France (BNF). [2]. Les manuscrits écrits en éthiopien sont issus des collections de Mondon-Vidailhet, Griaule et Marcel Cohen. Ceux en Syrique sont issus des collections de l'Ancien Testament dans la version peshitta. Ceux en Grec sont issus de la collection ayant appartenu au chancelier Séguier. Ceux en latin sont issus de Gregorius Magnus, Homiliae in Ezechielem et d'un lectionnaire romain composé vers 670-680.

Les images des documents téléchargés ont été convertis du canal RGB (i.e. Red Green Blue) en des images niveaux de gris, c'est-à-dire vers un canal à une seule valeur. Téléchargées en `jpeg`, les images ont été converties dans le format `png` d'une taille moyenne de 300 `ko`. La taille des pages est de 1024×1400 pixels. Une description du jeu de données est décrite dans la Table 7.1.

Pour chaque script (i.e. Éthiopien, Syriaque, Grec et Latin) nous avons choisi au hasard deux manuscrits, mis en gras dans la table 7.1. Notre *vérité terrain* [3] (Ground Truth en anglais) est composée d'un total de 2523 pages étiquetées chacune avec le

2. http://www.bnf.fr/fr/collections_et_services/catalogues.html

3. Une *vérité terrain* est un ensemble d'images étiquetées par les différentes classes d'objets que nous souhaitons reconnaître en utilisant un algorithme d'apprentissage. L'étiquetage peut être fait par des experts ou automatiquement par une analyse des images.

script qui lui correspond. La *vérité terrain* est divisée en un jeu d'entraînement, en un jeu de validation pour l'optimisation des paramètres des différents algorithmes, et en un jeu de test pour valider les résultats obtenus par les algorithmes présentés. En se basant sur les bonnes pratiques tirées de l'état de l'art [Krig 2014], les proportions des trois jeux de données sont les suivantes : 60% pour l'entraînement, 25% pour la validation et 15% pour le test. La *vérité terrain* construite contient des images découpées dans des positions aléatoires de l'image, avec une dimension de 256×256. Nous n'avons pas effectué de traitement ni de déformation sur les données, ce qui veut dire que nous avons également gardé le type d'images données en exemple dans la figure 7.2 (c'est-à-dire des images dégradées ou contenant divers artefacts).

7.1.3 Notre approche pour la classification des scripts

Nous avons considéré deux principales méthodes pour l'extraction de caractéristiques. Nous nous référons à la première méthode comme une extraction faite main des caractéristiques, en anglais *handcrafted features method*. Nous nous référons à la deuxième méthode comme une extraction des caractéristiques apprises, en anglais *learned features method*.

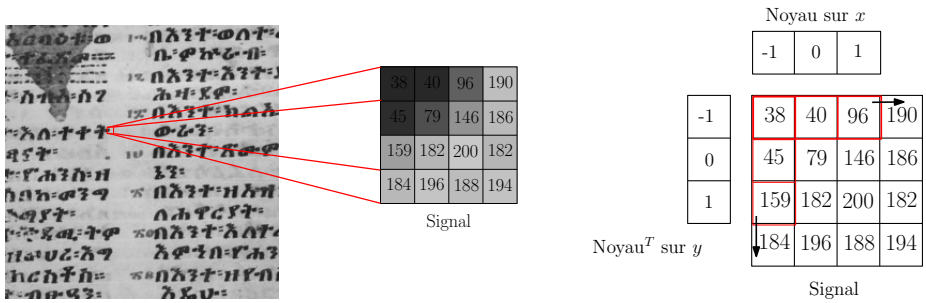
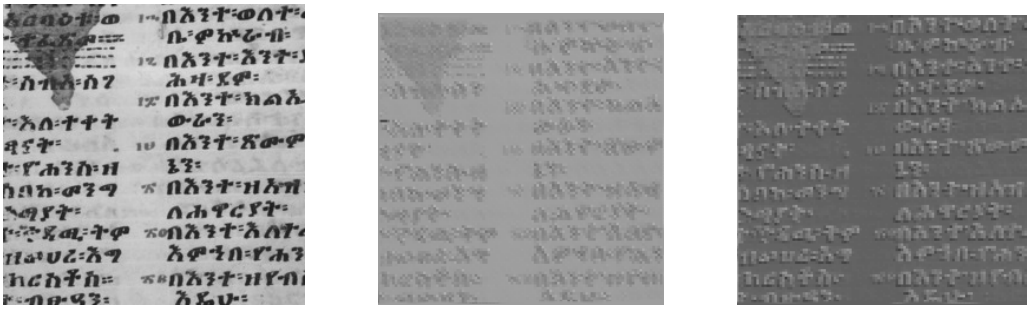


FIGURE 7.3 – (a) Extraction du signal au niveau du pixel de chaque image. (b) Application successive du masque $[-1, 0, 1]$ sur l'axe des x et $[-1, 0, 1]^T$ sur l'axe des y.

Comme dans tous les problèmes de traitement d'images, la page du manuscrit numérisé est représentée par une matrice de pixels. Nous nous référons à cette matrice comme étant un signal. L'extraction des caractéristiques à partir d'une image passe par la transformation du signal de cette dernière, usuellement représenté par une matrice de pixel, par exemple celle de la figure 7.3. La transformation s'effectue par une application d'une fonction de pondération appelée aussi masque. En pratique ceci revient à appliquer par translation sur la matrice de l'image un vecteur. Par exemple, le vecteur $[-1, 0, 1]$ traduit la différence entre la valeur du pixel d'avant et la valeur du pixel suivant. Comme c'est montré dans la figure 7.3. Un masque est appliqué sur une surface de pixels qui va être utilisée pour un certain traitement.

L'application de $[-1, 0, 1]$ sur l'axe x et de $[-1, 0, 1]^T$ sur l'axe y de l'image nous permet de calculer la dérivée du signal de l'image d'origine et de capturer

les changements au niveau des pixels. De grands changements nous permettront de capturer des traits ou de calculer les gradients dans une image. Notons que le processus que nous venons de décrire est commun à plusieurs techniques d'analyse des images nous y référons comme l'action d'appliquer un masque. Notamment lors de l'utilisation des réseaux de neurones artificiels, nous parlerons de filtre convolutif ou encore appliquer une convolution sur le signal de l'image. [4].



(a) (b) (c)

FIGURE 7.4 – Exemple de l'application du masque $[-1, 0, 1]$ sur l'axe x de l'image (a), obtention des caractéristiques représentées dans l'image (b). Application du masque $[-1, 0, 1]^T$ et obtention des caractéristiques représentées dans l'image (c).

Dans la figure 7.4, nous avons appliqué le masque $[-1, 0, 1]$ sur l'axe x de l'image (a), nous avons obtenu l'image (b). Ensuite, nous avons appliqué sa transposée sur l'axe y de l'image (a) et nous avons obtenu l'image (c). Ces caractéristiques obtenues sont qualifiées de faites main, car le masque $[-1, 0, 1]$ est déjà prédéfini. Les caractéristiques sont donc rigoureusement choisies par un expert pour y être ensuite remises en entrée dans un algorithme de discrimination ou classification. Par exemple, dans le domaine d'analyse des documents à partir d'images, nous pouvons citer les caractéristiques de Gabor [Hamamoto 1998], la technique du motif binaire local [Mu 2008], l'histogramme de gradients orientés [Dalal 2005], la transformation de caractéristiques visuelles invariante à l'échelle plus connue sous l'acronyme SIFT [Lowe 2004], etc.

L'approche que nous proposons d'étudier consiste à discriminer les scripts avec des caractéristiques apprises au terme d'une suite de convolutions successives appliquées sur les pixels au niveau de la **page** qui constituera notre ROI. Une convolution est une fonction qui est appliquée au signal, elle joue le rôle de filtre qui permet de distinguer certaines caractéristiques de ce signal par rapport à d'autres. Nous utiliserons une suite de convolutions, c'est-à-dire des convolutions de convolutions. Cette suite est structurée dans un ensemble de couches, l'ensemble de la structure est appelé architecture profonde par référence à l'approche communément appelée *Deep Learning* (voir la section 7.2 pour plus de détails).

L'objectif de l'utilisation de ces architectures profondes consiste à extraire des

4. En pratique une convolution est décrite par une matrice que l'on applique sur le signal (ou la matrice de pixels de l'image) comme c'est montré dans la figure 7.3

caractéristiques invisibles à l'oeil humain, mais interprétable et utiles pour les algorithmes de classification. Ces caractéristiques peuvent correspondre à des informations diverses telles que la courbure des traits, la forme des mots. Mais aussi d'autres informations qui sont considérées, par les autres méthodes, comme un bruit, par exemple, les informations qui concernent l'arrière-plan, les images, et d'autres artefacts présents dans les manuscrits. Ces bruits sont souvent supprimés dans des procédures de prétraitement. Nous pensons qu'un environnement non contrôlé avec un minimum d'intervention sur le jeu de données permettrait de réduire le biais et d'avoir une meilleure généralisation sur d'autres jeux de données [Torralba 2011].

Cette approche ne requiert pas d'expertise préalable sur les images et permet d'obtenir une représentation différente, mais satisfaisante comparée à l'extraction faite main. Comme pour l'apprentissage automatique en général, l'apprentissage des caractéristiques peut être divisé en deux sous catégories : un apprentissage non supervisé des caractéristiques et un apprentissage supervisé. Nous utiliserons respectivement les deux types d'apprentissages dans la section 7.4 et dans la section 7.5.

L'apprentissage non supervisé des caractéristiques ne requiert pas un étiquetage préalable des données d'entraînement. Nous pouvons apprendre et extraire des caractéristiques d'une image en utilisant *k-means* [Coates 2011], l'analyse en composantes principales (PCA) [Dorko 2003], les auto-encodeurs [Chen 2015, Wei 2015], les machines de Boltzman [Sahu 2015], etc. En revanche, l'apprentissage supervisé des caractéristiques requiert l'utilisation de la *vérité terrain* où nous disposons d'un étiquetage rigoureux des données.

À travers notre étude empirique de différentes architectures profondes, présentées tout au long de cette partie, nous voulons étudier l'hypothèse qui stipule qu'une architecture profonde est capable de se construire une représentation des données à partir d'une quantité limitée d'exemples. Et qui de plus n'a pas subi de prétraitement.

En travaillant avec les architectures profondes, les problèmes auxquels nous nous confrontons sont : la perte du signal d'origine au cours des convolutions successives avec une mauvaise optimisation des algorithmes, la qualité des données disponibles, et le non-contrôle de ce qui peut être appris. Par exemple, l'algorithme peut se faire une représentation des données à travers les variations des contrastes dans l'image ou la largeur des traits dans le texte, etc.

Dans un autre registre, le défi avec les architectures profondes comme d'autres algorithmes d'apprentissage automatique est la réduction du coût relatif aux erreurs commises lors de l'apprentissage (e.g. une instance mal classifiée). En particulier, ceci revient à optimiser une fonction objectif non convexe dans un espace de paramètres à plusieurs solutions. La différence entre une fonction convexe et non convexe est illustrée en une dimension dans la figure 7.5 (a) et 7.5 (b) respectivement. Dans la première, toute ligne reliant deux points sur (a) sera toujours au-dessus du minimum global ou de la solution optimale, tandis que pour la deuxième, toute ligne reliant deux points sur (b) peut être en dessous du minimum local.

Tandis que la recherche du minimum global d'une fonction convexe est un pro-

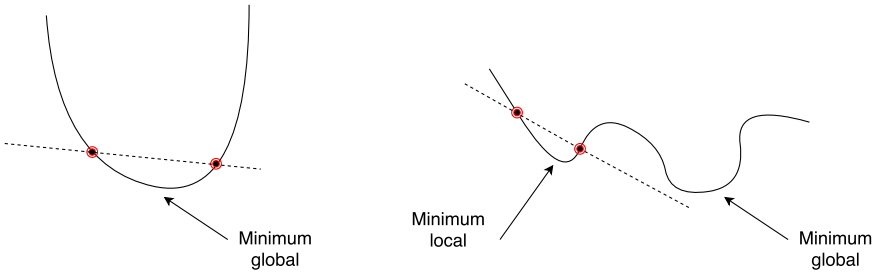


FIGURE 7.5 – (a) Illustration d'une fonction convexe avec une seule solution représentée par un minimum global. (b) Illustration d'une fonction non convexe avec plusieurs solutions, dont un minimum global.

blème résolu théoriquement, la recherche d'un minimum global d'une fonction non convexe est sujette à l'application d'une heuristique qui parcourt l'espace sans être sûre de trouver la solution optimale. Car, elle peut être prise au piège dans un minimum local et ne plus jamais en sortir. L'approche d'un tel problème consiste à explorer, empiriquement, l'espace des solutions en entreprenant plusieurs expérimentations, expliquées au travers des trois sections centrales de ce chapitre.

7.2 Méthode de travail

Les réseaux de neurones artificiels (ANN pour *Artificial Neural Network*) ont vu leur popularité augmenter au cours des dernières années. Tandis que l'implémentation des ANN a connu un succès modéré à leurs débuts dans les années 80, l'avancée technologique et l'arrivée des processeurs sur les cartes graphiques ont redonné à ses techniques une place importante. Sur certaines tâches telles que la reconnaissance d'objets, les ANN ont gagné une position difficile à détrôner par les autres algorithmes d'apprentissage telles que les séparateurs à vastes marges (SVM) ou encore les forêts aléatoires. Comme le montre l'étude de [Schmidhuber 2015], les ANN ont subi de nombreuses modifications et évolution au fils des années afin qu'ils soient adaptés à certaines tâches plus qu'à d'autres.

Nous distinguons deux grands groupes de ANN, les réseaux de neurones à convolutions (CNN pour *Convolutional Neural Network*) qui ont fait leurs preuves essentiellement en vision et en reconnaissance d'objet. Et les réseaux de neurones récurrents (RNN pour *Reccurent Neural Network*), qui sont utilisés pour apprendre une séquence de données par exemple la suite de caractères dans un texte numérique. Les CNN appliquent un filtre (décrit par une fonction) sur les données pour faire émerger des caractéristiques utiles à la classification. Les RNN donnent en sortie une combinaison des transformations linéaire appliquée sur les données en entrée, l'apprentissage de la séquence se fait par l'intégration d'un cycle dans la structure du réseau.

Nous nous intéressons dans cette partie à l'extraction des caractéristiques par

l'application d'un filtre appris. Nous utiliserons pour cela deux types de CNN, le premier appelé auto-encodeur entreprend un apprentissage non supervisé du filtre. Le deuxième est un CNN classique, il entreprend un apprentissage supervisé du filtre. Les auto-encodeurs sont des réseaux de neurones qui sont entraînés pour fournir une représentation alternative des données x qu'ils reçoivent en entrée, ceci est effectué par l'application d'une suite de convolutions, ensuite une suite de dé-convolutions.

Nous nous référons à la structure de dispositions des couches de convolutions et les couches de classifications comme une architecture. Nous appellerons architecture profonde, une architecture avec plusieurs couches de convolutions ou couches d'auto-encodeurs. L'apport méthodologique dans cette partie consiste en l'exploration empirique des architectures que nous avons introduites pour la tâche de l'identification des scripts dans les manuscrits anciens. Dans ce qui suit, nous commencerons par un rappel sur les ANN et introduirons par la suite les outils méthodologiques utilisés dans cette partie.

7.2.1 Rappels sur les réseaux de neurones artificiels

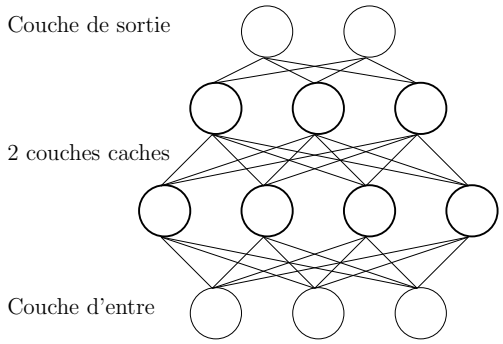


FIGURE 7.6 – Exemple d'un réseau de neurones artificiel avec une couche d'entrée, deux couches cachées et une couche de sortie. Nous avons illustré des liens bi directionnels entre les noeuds pour illustrer le sens de la convolution et le sens de l'algorithme de rétropropagation.

Le réseau de neurones artificiels (ANN), ou perceptron multicouche (ANN), peut être défini comme un ensemble d'unités de traitement simples (neurones) qui sont reliées entre elles par des connexions pondérées. Dans la figure 7.6, le réseau de neurones artificiels est constitué :

— d'une couche d'entrée de dimension $d = 3$. La valeur d dépend du vecteur d'observation reçu par la couche d'entrée.
— Ensuite de deux couches composées d'unités cachées qui appliquent une fonction non linéaire sur les transformations linéaires qu'elles reçoivent de la couche d'entrée.
— Et enfin, la couche de sortie, de dimension m.

Dans la figure 7.6, $m = 2$. Dans ce cas, ce réseau, cité en exemple, est utilisé pour une tâche de classification binaire. Pour d'autres valeurs de m, le réseau sera

utilisé pour une classification multi classes. Dans la couche de sortie, le neurone avec la valeur la plus élevée donnera la classe.

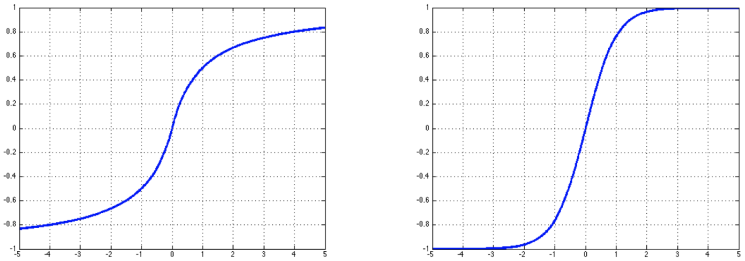
Afin de simplifier l'explication, prenons le cas d'un réseau de neurones à une seule couche cachée comportant d_h unités cachées [5]. La sortie du réseau est obtenue par le calcul de la fonction $f : \mathbb{R}^d \rightarrow \mathbb{R}^m$, où d est la taille du vecteur d'entrée x, et m la taille du vecteur de sortie $f(x)$, telle que :

$$f(x) = g(b_2 + W_2(h(b_1 + W_1 x))) \tag{7.1}$$

$b_1 \in \mathbb{R}^{d_h}$ et $b_2 \in \mathbb{R}^m$ sont les vecteurs de biais. $W_1 \in \mathbb{R}^{d_h \times d}$ représente la matrice des poids connectant le vecteur d'entrée à la couche cachée, $W_2 \in \mathbb{R}^{m \times d_h}$ est la matrice des poids connectant la couche cachée au vecteur de sortie. La fonction 7.1, calcul la sortie du neurone artificiel vers l'environnement extérieur par l'application de la fonction d'activation h sur la matrice des poids en entrée $W1$ et l'application de la fonction d'activation g sur la sortie de h qui est passée aux neurones de la couche cachée. L'interprétation de la sortie de cette fonction 7.1 dépend du problème de classification considéré. Par exemple dans le cas binaire 1 signifierait que l'entré appartient à une classe A et 0 signifierait que l'entrée appartienne à la classe B. Les fonctions d'activation g et h sont définies comme suit : typiquement pour h l'on choisit la fonction sigmoïde ou la fonction tangente hyperbolique :

$$sigmoide(a) = \frac{1}{1 + e^{-a}} : [-\infty, +\infty] \rightarrow [0, 1] \tag{7.2}$$

$$tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} : [-\infty, +\infty] \rightarrow [-1, 1] \tag{7.3}$$



(a) (b)

FIGURE 7.7 – Illustration des réponses non linéaires des fonctions d'activations *softsign* (a) et *than*(b)

Pour la discrimination des scripts avec des réseaux de neurones, nous utiliserons la fonction *tanh* dans nos travaux, car elle permet un apprentissage plus rapide en raison de sa saturation en 1. Dans le cas de l'apprentissage des caractéristiques, comme nous le verrons plus tard, nous utiliserons la fonction *softsign*. Comme nous pouvons observer dans la figure 7.7, la dérivée de la fonction *softsign* illustrée en (a)

5. d_h dimension des couches cachées, h est utilisée pour *hidden* : cachée

converge polynomialement vers 0, ce qui est plus intéressant pour la phase d'apprentissage qu'une dérivée de fonction qui converge de façon exponentielle et sature en 1 rapidement comme la fonction *tanh* illustrée en (b). La première raison de ce choix revient à la nature de nos données qui sont bruitées et en quantité limitée qui nous a poussés à penser qu'un apprentissage des caractéristiques qui ne converge pas rapidement pourrait apporter plus d'informations. La seconde raison de ce choix porte sur la rapidité de cette fonction d'activation lors de l'apprentissage des caractéristiques. La fonction *softsign* est définie comme suit :

$$softsign(x) = \frac{x}{|x|+1} : [-\infty, +\infty] \rightarrow [-1, 1] \tag{7.4}$$

On peut réécrire le vecteur de sortie $f(x)$ dans la fonction 3.12 comme suit :

$$f(x) = g(c + V(h(x))) \tag{7.5}$$

Dans le cas d'une classification multi classes, le choix de g se porte le plus souvent sur la fonction $softmax$ décrite comme suit :

$$softmax(a)_i = \frac{e^{a_i}}{\sum_j e^{a_j}} \tag{7.6}$$

Algorithme 13 Apprentissage des paramètres d'un réseau de neurones artificiels avec la descente de gradient

```
1: Initialiser aléatoirement les poids du réseau
2: tant que condition d'arrêt n'est pas atteinte faire
3:     pour tout exemple dans l'ensemble d'entraînement faire
4:         Calculer le gradient de la fonction de perte L
5:         Appliquer la technique de rétropropagation du gradient
6:         Mettre à jour les poids W du réseau
```

Pour entraîner un réseau ANN, nous devons apprendre tous les paramètres $\theta = \{W, V, b, c\}$. Pour cela, nous avons utilisé dans nos travaux l'algorithme de descente de gradient (voir pseudo-code dans Algorithme 13). L'algorithme est divisé en deux étapes : une propagation en avant pour calculer le vecteur de sortie $f(x)$ et une propagation en arrière où la dérivée partielle de la fonction de perte est calculée par rapport aux paramètres du réseau. Ceci est décrit dans la fonction suivante :

$$W \leftarrow W - \eta \frac{\partial L}{\partial W} \tag{7.7}$$

Où η est un pas d'apprentissage, et $\frac{\partial L}{\partial W}$ le gradient de la perte L par rapport aux paramètres W.

Un réseau ANNs ayant suffisamment d'unités cachées est en mesure d'apprendre à représenter n'importe quelle fonction continue. Dans ce cas, il est considéré comme un approximateur universel [Hornik 1989]. Cependant, le nombre de paramètres à apprendre augmente de façon exponentielle avec le nombre de couches cachées [Hastad 1991] ce qui nous empêche d'avoir un temps d'apprentissage "souhaitable"

sur des architectures multicouches [Bengio 2007c]. De nos jours, grâce aux nouvelles architectures, fonctions d'activations et méthodes d'entraînement combinées à l'augmentation de la puissance des ordinateurs, l'utilisation des architectures dites profondes est rendue possible. Nous nous concentrerons dans cette partie sur l'utilisation des réseaux de neurones à convolution pour l'extraction des caractéristiques apprises.

7.2.2 Les réseaux de neurones à convolutions (CNN)

Les réseaux de neurones à convolution (CNN pour *Convolutional Neural Networks*) sont inspirés des travaux de Hubel sur le cortex visuel des mammifères et l'arrangement des cellules responsables de l'interprétation du champ visuel [Hubel 1968]. On distingue essentiellement deux types de cellules. Les cellules dites simples, ils détectent les caractéristiques liées à la forme et les cellules dites complexes qui sont sensibles à la forme entière.

Les CNNs ont été popularisées avec les travaux de [Lecun 1998a] et son implémentation appelée LeNet5 (5 pour les 5 régions de l'aire visuelle), pour la reconnaissance de caractères. Cependant, les CNNs trouvent leur origine dans les travaux de [Fukushima 1980] sur la simulation du fonctionnement des cellules dans cortex visuel des mammifères.

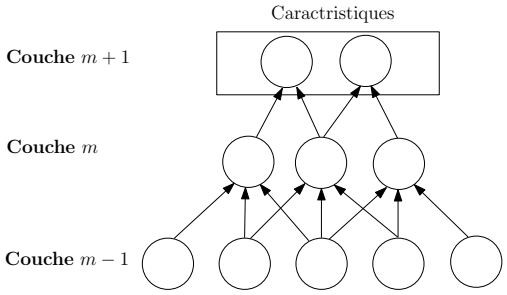


FIGURE 7.8 – Caractéristiques/Poids partagées

Dans les CNNs, chaque neurone est connecté à un sous-ensemble de la couche précédente. Dans la figure 7.8, la couche $m-1$ comporte 5 neurones, les trois neurones de la couche m sont connectés à trois neurones voisins de la couche $m-1$. On peut considérer chaque neurone comme une unité de détection d'une caractéristique locale. La carte des caractéristiques (*features map*)[6] est construite par convolution c'est-à-dire par application d'un filtre linéaire dont le principe consiste à utiliser une fenêtre glissante pour parcourir les données. Comme c'est décrit dans l'équation 7.2.2.1, chaque connexion entre les neurones est pondérée par une matrice des poids W et un vecteur de biais b qui contrôle le potentiel d'activation des neurones. La sortie est obtenue par l'application d'une transformation non linéaire g. Le passage

6. Ensemble de neurones qui sont considérés comme des unités de détection des caractéristiques locales.

d'une couche à une autre a pour effet de réduire le signal global en des caractéristiques locales.

$$h_{ij}^{k} = g((W^{k} * x)_{ij} + b_{k}) \tag{7.8}$$

L'architecture du réseau construit avec des convolutions impose le partage des poids pour toute carte k de caractéristiques. Car, les neurones calcul la même fonction avec des entrées différentes provenant de différentes parties du système, dont il faut qu'ils aient le même poids. Ce principe de partage de poids réduit considérablement la complexité du modèle et facilite également la généralisation, puisque l'apprentissage se fait avec les mêmes paramètres pour extraire les mêmes caractéristiques des différentes parties du signal global.

7.2.2.1 Couche de *Max-Pooling* ou max-agrégation

Enfin, un autre concept important des CNNs est le sous-échantillonnage appliqué sur la fenêtre glissante de la convolution. Appelé max-pooling ou max-agrégation, il a pour effet de ne prendre que la valeur maximale de la sortie de la fonction d'activation g appliquée sur différentes régions, qui ne se chevauchent pas, du signal global. De la même manière que l'équation , un vecteur de biais est rajouté aux résultats en sortie. Le max-pooling permet de réduire la sensibilité aux translations de la fenêtre et aux faibles rotations. Il de ce fait permet d'éviter aussi des calculs superflus.

7.2.3 Les auto-encodeurs (AE)

Les auto-encodeurs [Ranzato 2007, Bengio 2007a] sont des réseaux de neurones qui sont entraînés pour fournir une représentation alternative des données x qu'ils reçoivent en entrée. Un auto-encodeur qui aura appris la bonne représentation des données x est capable d'encoder puis de décoder de manière à retrouver approximativement les mêmes données x en entrée. L'intérêt d'utiliser les auto-encodeurs n'est pas dans le fait de retrouver x, mais dans la transformation qu'aura appris l'auto-encodeur pour arriver à ce résultat. C'est cette représentation apprise que nous avons fournie comme un pré entraînement au réseau, afin de remplacer l'initialisation aléatoire des connexions neuronales ou d'extraire de bonnes caractéristiques apprises.

Dans la figure 7.9, nous avons illustré un exemple d'auto-encodeur avec une seule couche. Le vecteur des caractéristiques apprises, à droite de la figure, est issu de la représentation que l'auto-encodeur a construite avec l'exécution d'une descente de gradient (voir l'algorithme 13) et ensuite l'exécution de l'algorithme de rétropropagation pour ajuster l'initialisation des connexions neuronales.

Le principe de fonctionnement de l'auto-encodeur consiste en l'application de deux tâches : l'encodage et le décodage. L'encodage est une transformation déterministe, que l'on note z, du vecteur en entrée, $x \in [0, 1]^d$. Notons que dans nos travaux, nous utiliserons un vecteur d'entrée $x \in [-1, 1]^d$.

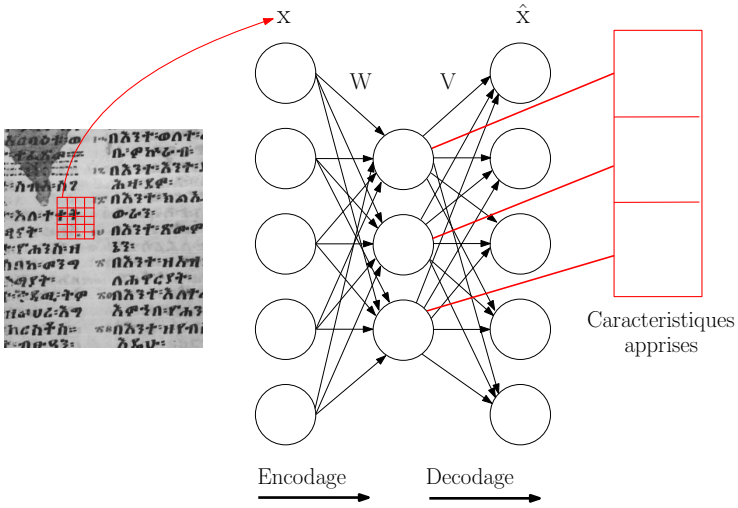


FIGURE 7.9 – Un auto-encodeur avec une seule couche cachée

$$z = f_{enc}(Wx + b) \tag{7.9}$$

Où W est une matrice $\in R^{d_h \times d}$ et $b \in R^{d_h}$ est un vecteur de biais. Le décodage est la reconstruction de la transformation h, que l'on note r.

$$\hat{x} = f_{dec}(Vz + c) \tag{7.10}$$

Où V est la transposée de W, s'il s'agit de matrices liées, et c un biais de dimension d. Comme pour un réseau de neurones, les paramètres sont $\theta = \{W, V, b, c\}$ et les poids sont $V = W^T$ s'il s'agit de matrices liées. Ces paramètres sont optimisés avec la méthode de descente de gradient qui a pour but de trouver un optimum global. Pour une entrée x binaire ou quasi binaire, il est préférable d'optimiser l'erreur de reconstruction par le calcul d'une entropie croisée. L'entropie croisée mesure la distance entre la sortie donnée par le réseau et la sortie attendue. L'entropie croisée est définie dans l'équation 7.11, quant à l'équation de la perte résultante, elle est définie dans l'équation 7.12.

$$L(x, \hat{x}) = -\sum_{i=1}^{d} [x_i \log(\hat{x}_i) + (1 - x_i) \log(1 - \hat{x}_i)] \tag{7.11}$$

$$L(x, \hat{x}) = \sum_{i=1}^{d} [(x_i - \hat{x}_i)^2] \tag{7.12}$$

7.2.4 Les auto-encodeurs convolutif empilés (SCAE)

Les auto-encodeurs peuvent être empilés, où la sortie d'un auto-encodeur est utilisée comme entrée pour l'auto-encodeur suivant. Cette utilisation séquentielle augmente le taux de compression des données en entrées et réduit le nombre de

caractéristiques dans la couche $l+1$ par rapport à la couche l. Cette structure peut être associée à un réseau de neurones à convolution pour avoir un vecteur de caractéristiques en plusieurs dimensions, ceci en appliquant le même auto-encodeur sur plusieurs grilles de pixels.

Dans cette thèse, et pour apprendre les caractéristiques, nous utiliserons un empilement d'auto-encodeurs avec l'application d'une convolution à chaque couche pour la transformation du signal de l'image d'origine. Les auto-encodeurs convolutifs empilés ou SCAE (Stacked convolutional autoencoders) sont un réseau de neurones avec plusieurs couches d'auto-encodeurs. Cet empilement d'auto-encodeurs forme une architecture profonde où le résultat de la couche l_n est repris par la couche l_{n+1}.

L'apprentissage non supervisé se fait sur cette structure en appliquant un algorithme glouton, une heuristique qui va chercher le minimum local pour chaque couche [Bengio 2007b]. Les poids du réseau sont affinés en utilisant la méthode de descente de gradient, le but est d'arriver à un minimum global pour minimiser le risque empirique. Le vecteur de caractéristique produit par la couche finale peut être donné en entrée à un modèle de classification analogue à un réseau de neurones à convolution afin d'initialiser ses poids.

En ce qui concerne la fonction d'activation des auto-encodeurs empilés, nous avons choisi de travailler avec la fonction *softsign*. L'auto-encodeur avec lequel nous avons travaillé encode une entrée x de dimension n en une sortie y de dimension m comme suit :

$$y_i = f\left(b_i^e + \sum_{j=1}^{n}\left(w_{ij}^e \cdot x_j\right)\right) \tag{7.13}$$

Où b_i^e est le biais et w^e les poids utilisés pour l'encodage.

Le décodage de la sortie y pour reconstruire l'entrée est exécuté d'une façon similaire :

$$\hat{x}_j = f\left(b_j^d + \sum_{i=1}^{m}\left(w_{ji}^d \cdot y_i\right)\right) \tag{7.14}$$

Où $\hat{x}$ est l'approximation du vecteur x encodé par y, b_j est le biais, et w^d sont les poids utilisés pour le décodage. Pour calculer la fonction 7.14, l'auto-encodeur a besoin d'apprendre un total de poids $m \times (n+1) + n \times (m+1)$, durant la phase d'entraînement. Pour optimiser le temps d'entraînement, il est plus judicieux d'utiliser des convolutions successives d'auto-encodeurs empilés avec des signaux de petite taille que d'entraîner, en une seule fois, un auto-encodeur qui couvre une grande portion de l'image ou un signal de grande taille. Car, rappelons que les auto-encodeurs donnent une représentation alternative à l'ensemble du signal. Tandis que, les piles d'auto-encodeurs fournissent des représentations alternatives de plusieurs ordres. C'est-à-dire de l'ordre des traits, contours, etc. Jusqu'à arriver à l'ordre de l'objet en entier.

Les convolutions sont créées comme suit : d'abord, un auto-encodeur couvrant une surface de $W_1 \times H_1$ pixels avec un nombre de sortie m_1 est

entraîné. Ensuite, nous le copions $W_2 \times H_2$ fois, et rajoutons les copies dans une grille avec un décalage de $O_{1x} \times O_{1y}$ pixels. La grille couvre alors, $((O_{1x} \cdot (W_2 - 1) + W_1) \times (O_{1y} \cdot (H_2 - 1) + H_1))$ pixels. La sortie des auto-encodeurs dans cette grille peut être considérée comme un tableau composé de $W_2 \times H_2 \times m_1$ valeurs. Ce tableau peut être par la suite donné en entrée à un auto-encodeur dans un second niveau dans la convolution.

Notons qu'il faut faire attention à la dimension des convolutions pour pouvoir empiler plusieurs niveaux d'auto-encodeurs. Enfin, tous les niveaux des auto-encodeurs empilés avec une convolution sont entraînés avec l'algorithme de descente de gradient (voir algorithme 13) et une rétropropagation sur les poids, le but étant de minimiser $(\hat{x} - x)^2$.

Les poids des connexions neuronales ont été initialisés avec une distribution uniforme telle que

$$W \in]-\frac{1}{\sqrt{n_{in}}}; \frac{1}{\sqrt{n_{in}}}[\tag{7.15}$$

Où n_{in} est le nombre de neurones en entrée. La valeur des neurones en sortie est dans l'intervalle $[-1, 1]$. Le réseau avec lequel nous travaillons n'est pas *Sparse*, ou en d'autres termes, il ne contient pas de zéros. À quantité égale d'informations encodées, ce type de réseau possède une phase d'entraînement plus rapide que les réseaux *Sparse*.

7.2.4.1 Les couches d'*Extremum Pooler* : concept additionnel

Nous appelons *Extremum Pooler* une opération de réduction sur un réseau dense (i.e. non *Sparse*) par la sélection exclusive des neurones avec la valeur maximale. Cette fonction est similaire au *Maxpooling* qui est exécutable sur un réseau *Sparse*. La fonction d'*Extremum Pooler* est définie par

$$a_j = \max_{n \times n}(a_i^{m \times m} J(m, m)) \tag{7.16}$$

où $J(m, m)$ est une fenêtre glissante sur le *patch* en sortie d'une couche d'un SCAE ou d'une convolution. J calcule la valeur maximale dans le voisinage défini par la taille de la fenêtre glissante sur le *patch* d'une taille $m \times m$ pixels.

7.2.5 Architectures

Dans le contexte de ce travail, nous appelons une architecture une structure d'interconnexion entre différentes couches de convolutions, de max-pooling, d'extremum-pooling ou d'auto-encodeurs. Une architecture comprend dans son entrée une image et en sa sortie un vecteur de caractéristiques ou un réseau de neurones pour la classification. Par exemple, la figure 7.15 est une suite d'auto-encodeurs empilés qui transforme l'image en entrée en un vecteur de caractéristiques en appliquant différents filtres. Chaque filtre réduit la taille du signal d'origine et possède plusieurs

sorties. Par exemple, l'application du premier filtre dans la figure 7.15 résulte en 8 images de taille 26×26.

Au vu de la complexité de nos données, nous explorons dans ce travail plusieurs architectures dans le but d'obtenir une bonne taille du filtre appris. Car, il faut qu'il soit assez large pour capturer les variations des différents traits des scripts et assez petit pour ne pas capturer trop de variations et perdre l'information essentielle à la classification.

7.3 Construction d'un référentiel empirique obtenu

Nous introduisons dans cette section notre référentiel empirique. Ce référentiel est élaboré avec trois algorithmes qui sont entraînés avec des caractéristiques extraites avec l'histogramme de gradients orientés (HOG en anglais, *Histogram of Oriented Gradients*). Ce référentiel a pour but de comparer les solutions à base d'architectures profondes que nous proposons.

7.3.1 Histogramme de gradients orientés

FIGURE 7.10 – Détection des piétons avec des histogrammes de gradients orientés

Dans le référentiel empirique, les algorithmes de classification sont entraînés avec des caractéristiques extraites avec un filtre fait main, en anglais *handcrafted features*. Nous avons choisi d'utiliser l'histogramme de gradients orientés (HOG) qui a été proposé par [Dalal 2005] pour détecter les piétons dans une image par exemple dans la figure 7.10. Les caractéristiques de type HOG ont également été utilisées par [Minetto 2013] avec succès pour la reconnaissance des textes dans les images et par [Yilmaz 2011] pour la reconnaissance du tracé d'une signature. Étant donné la nature complexe des images de manuscrits, notre choix s'est porté sur l'extraction des caractéristiques avec HOG. Car, il est plus résistant aux transformations géométriques et photométriques dans les images.

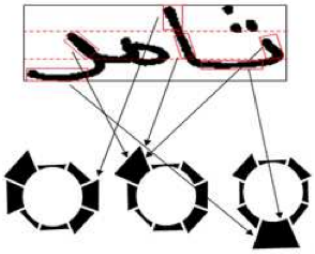


FIGURE 7.11 – Des histogrammes de gradients orientés divisés en 8 *bins*. Chacun des trois histogrammes correspond à la partie supérieure, celle du milieu et la partie inférieure de l'image du script arabe, donné ici en exemple. La flèche indique la contribution d'une partie spécifique du tracé à l'histogramme [Saidani 2015]

HOG caractérise une image par une distribution de gradients. Chaque gradient représente un changement de direction dans l'intensité du pixel ou la couleur dans une image. Les caractéristiques HOG sont représentées par un histogramme qui compte l'orientation du gradient des pixels d'une image donnée. Le principe est le suivant : d'abord les orientations du gradient de chaque pixel sont calculées, puis un histogramme des orientations est calculé sur une petite portion de l'image. Enfin, tous les histogrammes de toutes les portions sont rassemblés dans un vecteur. Ce vecteur est ce que l'on appelle vecteur de caractéristiques et il sera donné en entrée aux algorithmes de classification du référentiel empirique. Ce principe est décrit en détail dans ce qui suit.

7.3.1.1 Extraction des caractéristiques

En pratique, le vecteur des caractéristiques HOG est extrait comme suit : d'abord, l'image est divisée en des petites portions (e.g. 6×6 pixels), où l'on applique les filtres $D_x = [-1, 0, 1]$ et $D_y = [-1, 0, 1]^T$, respectivement sur l'axe x et y de l'image I. Cette opération de convolution, comme ça a été montré précédemment dans la figure 7.3, nous permet d'obtenir les dérivées $\delta_x = I \times D_x$ et $\delta_y = I \times D_y$, comme ça été également montré dans la figure 7.4 où δ_x correspond à la figure (a) 7.4 et δ_y correspond à la figure (b) 7.4.

Puis, la magnitude et l'orientation du gradient sont calculées avec les équations 7.17 et 7.18.

$$Magnitude_{x,y} = \sqrt{(\delta y_{x,y})^2 + (\delta x_{x,y})^2} \tag{7.17}$$

$$Angle_{x,y} = atan(\delta y_{x,y}, \delta x_{x,y}) \tag{7.18}$$

Ensuite, un histogramme des orientations est calculé pour chaque portion en utilisant l'équation suivante :

$$H_i = \sum_{x,y \in I, Angle_{x,y} \in bin_i} Magnitude_{x,y}, i = 1, \ldots, n \tag{7.19}$$

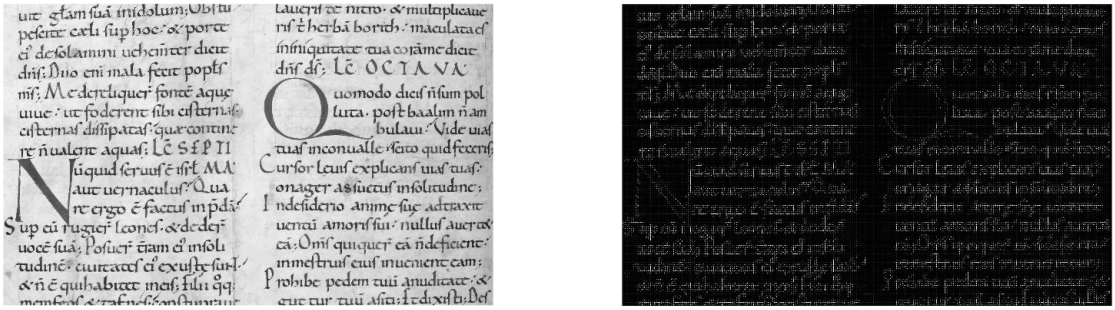


FIGURE 7.12 – Calcul de l'histogramme des gradients orientés sur une image d'une page d'un manuscrit écrit en latin.

Suivant les gradients calculés, les histogrammes sont uniformes de 0 à 180° (cas non signé) ou de 0 à 360° (cas signé). Dans nos expérimentations, l'intervalle $[0, 2\pi]$ (i.e. 0 à 360°) est divisé en ce que l'on appelle des *bins*, comme c'est montré dans la figure 7.11. Chaque *bin* couvre une orientation de $\frac{\pi}{4}$. Ce qui nous donne un n = 8 bins. Ce choix est expliqué par les expérimentations précédentes menées par [Saidani 2015] sur la reconnaissance du tapé et du tracé des mots dans une image.

Chaque pixel de la portion vote pour un *bin* de l'histogramme, en fonction de l'orientation du gradient au point (x,y). Le vote du pixel est pondéré par l'intensité du gradient en ce point.

Enfin, les gradients de tous les blocs sont normalisés dans un histogramme et régularisés avec la norme L_2 décrite par l'équation 7.20, où les histogrammes de chaque bloc sont désigné par v.

$$f = \frac{v}{\sqrt{||v||_2^2 + \varepsilon^2}} \tag{7.20}$$

Une fois normalisées, les valeurs de l'histogramme sont collectées dans un vecteur de caractéristiques et sont par la suite données en entrée aux algorithmes de classification présentée dans la sous-section suivante. La figure 7.12 illustre la sortie obtenue avec HOG après son application sur une page d'un manuscrit latin.

7.3.2 Classification

Pour construire notre référentiel de comparaison des différentes stratégies d'analyse des scripts, nous avons donné, en entrée, à trois différents algorithmes de classification les caractéristiques extraites avec HOG. Nous avons choisi le perceptron multicouche appelée aussi réseaux de neurones artificiels, les séparateurs à vaste marge et les forêts aléatoires.

7.3.2.1 Réseaux de neurones artificiels

Les réseaux de neurones artificiels sont constitués de plusieurs unités logiques appelées neurones, qui répondent que par 0 ou 1. Les neurones sont disposés dans trois différentes couches : la couche d'entrée, la couche cachée et la couche de sortie. La couche de sortie permet de diviser l'espace d'entrée de façon linéaire, les ou la couche cachée permet applique une division non linéaire sur l'espace. Pour plus de détails, veuillez vous référer à la section de rappel 7.2.1. Nos expérimentations ont été menées avec un nombre de couches cachées égales à $\frac{\#attributs+\#classes}{2}$. Par exemple si nous avons six attributs et les quatre classes correspondantes aux différents scripts alors le nombre de couches est égal à 5.

7.3.2.2 Forêts aléatoires

Les forêts aléatoires (RF pour Random Forest en anglais) introduites par [Breiman 2001] sont un ensemble d'arbres de décisions entraînés aléatoirement. Chaque arbre produit une prédiction et l'ensemble des prédictions est ensuite agrégé avec différentes techniques [Breiman 2001], afin de produire un résultat de prédiction final. Pour plus de détails, veuillez vous référer à la section 5.2.

7.3.2.3 Séparateurs à vaste marge

Les séparateurs à vastes marges (SVM) ont été introduits par [Vapnik 1998]. La méthode consiste à chercher un hyperplan qui sépare deux classes de données de manière à maximiser la marge entre les exemples les plus proches des deux différentes classes. On se réfère à ses exemples comme des vecteurs de support. Pour plus de détails, veuillez vous référer à la section de l'annexe A.2.3.

7.3.3 Résultats

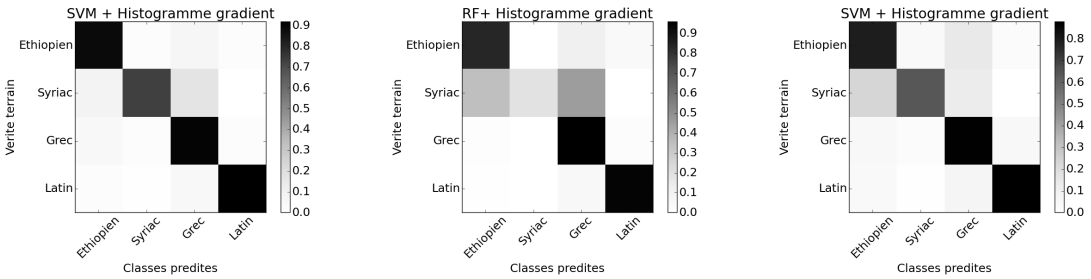
Afin de juger la fiabilité des trois algorithmes de classification (i.e. ANN, RF, SVM) entraînée avec les caractéristiques HOG, nous avons utilisé la validation croisée sur une base de 2523 images de pages de manuscrits. Nous avons fixé le nombre de portions (*folds*) à 10 pour chaque itération, c'est-à-dire que l'évaluation de l'erreur est calculée sur 10 sous-échantillons tirés de la base des images.

Les résultats sont rapportés dans la table 7.2 où nous pouvons constater que les réseaux de neurones artificiels ont obtenu le meilleur résultat avec un taux de justesse égal à 89.07% suivi des forêts aléatoires avec un taux de justesse égale à 86.15%. Cependant, selon la matrice de confusion illustrée dans la figure (b) 7.13, nous constatons que les forêts aléatoires n'arrivent pas à généraliser leur apprentissage pour discriminer le script syriaque.

L'algorithme SVM paraît plus résistant à la quantité de données fournies lors de l'apprentissage et arrive à reconnaître les quatre scripts malgré une performance moindre par rapport aux deux autres. Notons que la quantité de données pour ce

HOG		Precision moy.	Rappel moy.	F1 moy.	Justesse	RMSE
ANN	Ethiopien	0.86	0.88	0.87	**89.07%**	0.12
	Syriaque	0.78	0.73	0.75		
	Grec	0.87	0.90	0.88		
	Latin	**0.95**	**0.92**	**0.93**		
RF	Ethiopien	0.85	0.83	0.84	86.15%	0.15
	Syriaque	0.96	0.19	0.32		
	Grec	0.80	0.96	0.87		
	Latin	**0.94**	**0.94**	**0.94**		
SVM	Ethiopien	0.79	0.79	0.79	83.70%	0.18
	Syriaque	0.72	0.64	0.67		
	Grec	0.81	0.87	0.84		
	Latin	**0.92**	**0.88**	**0.90**		

TABLE 7.2 – Les résultats des algorithmes de discrimination entraînée avec les caractéristiques issues de l'histogramme de gradients orientés. Nous avons mené un ensemble de 10 tests sur les données jamais vues auparavant par les algorithmes de discrimination. ANN est l'abréviation pour le réseau de neurones artificiels, RF est l'abréviation des forêts aléatoires, et SVM est l'algorithme de machine à vecteurs de support.



(a) (b) (c)

FIGURE 7.13 – Matrice de confusion des algorithmes de classification avec les caractéristiques extraites avec l'histogramme de gradient orienté. La figure (a) illustre les résultats de discrimination du ANN. La figure (b) illustre les résultats de discrimination des forêts aléatoires. Enfin, la figure (c) illustre les résultats de discrimination exécutés par SVM. Les meilleurs résultats sont mis en gras.

script est réduite par rapport aux trois autres, ce qui explique notamment la faible performance des trois algorithmes pour la reconnaissance du Syriaque en particulier.

7.4 Contribution pour l'apprentissage non supervisé de scripts avec les auto-encodeurs

Dans cette section, nous étudions l'apprentissage non supervisé des caractéristiques pour la reconnaissance des scripts. L'intérêt pour l'apprentissage non supervisé des caractéristiques a été porté par [Hinton 2006a, Bengio 2007b, Huang 2007, Coates 2010, LeCun 2012]. Il est présenté comme une alternative logique des ca-

ractéristiques extraites par des filtres choisis ou pré fabriqué par des experts (i.e. *handcrafted features*), et comme une technique qui exploite la puissance des réseaux de neurones à plusieurs couches, par rapport à l'apprentissage supervisé. Défendu donc par Lecun, Bengio, Hinton et leurs étudiants, l'apprentissage non supervisé des caractéristiques a été récemment introduit dans l'analyse des images des documents.

Nous citons par exemple le travail de [Sahu 2015] sur la reconnaissance des caractères à partir d'une image d'un document avec un ROI au niveau des mots. Les auteurs ont utilisé un empilement de machines de Boltzman restrictives (RBM) pour obtenir des représentations compactes des images contenant du texte. Les RBMs sont des réseaux de neurones avec des unités stochastiques par comparaison aux unités déterministes comme la sigmoïde ou le softsign (voir le rappel sur les réseaux de neurones dans la sous-section 7.2.1). Les auteurs ont ensuite appliqué des algorithmes classiques des reconnaissances de caractères (OCR) sur les caractéristiques apprises par les RBMs et ont démontré de meilleures performances comparées aux caractéristiques extraites manuellement.

Pour notre travail, nous utilisons un empilement d'auto-encodeurs, principe présenté dans la sous-section 7.2.4. Un auto-encodeur, rappelons-le, est un réseau de neurones qui apprend à travers des convolutions successives une représentation des données en entrée. Cette représentation, si elle est bonne, doit permettre à l'auto-encodeur de recréer avec une suite de déconvolution le signal d'entrée (voir sous-section 7.2.3). Le principe est le même que les RBM sauf qu'un auto-encodeur utilise des unités déterministes et dans ce cas en particulier des unités de type *softsign* (voir le rappel de la sous-section 7.2.1).

L'extraction des caractéristiques avec un filtre appris par un empilement d'auto-encodeurs et leur utilisation pour la classification reste encore mal explorée dans l'état de l'art. Nous citons par exemple le travail de [Chen 2015] sur l'extraction et la sélection des caractéristiques apprises pour la tâche de l'analyse de la mise en page des manuscrits anciens. Les auteurs ont utilisé l'algorithme des séparateurs à vaste marge (SVM) entraîné avec les caractéristiques apprises. Les auteurs de [Wei 2015] ont utilisé la même architecture que [Chen 2015] et ont travaillé sur la sélection des caractéristiques apprises par l'utilisation de l'algorithme de *feed forward selection*.

7.4.1 Apprentissage des caractéristiques

Dans ce qui suit, nous présentons trois différentes architectures [7] avec différentes dispositions de SCAE. La première est un empilement classique d'auto-encodeurs convolutifs (SCAE), la deuxième est un empilement de SCAE avec une dernière couche d'*Extremum pooler* à la suite de la dernière couche SCAE, et la troisième est une architecture en Sandwich où chaque couche SCAE est suivie d'une couche d'*Extremum pooler*. Nous utilisons l'espace de travail *N-light-N* proposé par [Seuret 2016] [8], qui nous a permis d'implémenter les trois différentes architectures. À la sortie de chaque architecture, les caractéristiques apprises sont récupérées dans

7. Tel que c'est défini dans la sous-section 7.2.5.
8. https ://github.com/seuretm/n-light-n

un vecteur et utilisées pour entraîner les mêmes algorithmes proposés dans notre référentiel.

Dans la première architecture, nous souhaitons expérimenter l'effet sur le taux de justesse quand nous exécutons une rétropropagation sur les poids des SCAE. Dans la deuxième et troisième architecture, nous souhaitons expérimenter deux différentes dispositions des couches d'*Extremum pooler*. Ces dispositions sont inspirées des bonnes pratiques de l'état de l'art [Krizhevsky 2012, LeCun 2012]. Notons également qu'une rétropropagation sur des couches de sous-échantillonnage comme l'*Extremum pooler* est possible si nous pouvons garder une trace de tous les poids avant l'exécution d'un *Extremum pooler*. Ceci est en cours de développement dans l'espace de travail *N-light-N*.

Les images qui sont données en entrée aux différentes architectures sont d'une taille égale à $256 \times 256 \times 1$. Cette taille correspond à celle de la capture prise à un endroit aléatoire de la page d'un manuscrit appartenant aux données d'entraînement, le chiffre 1 correspond au canal gris, si les images été en RGB (i.e. Red, Green, Blue), nous aurions mis 3. Cette taille nous a permis d'obtenir un traitement plus rapide sur les processus CPU. Notons que nous avons également essayé des résolutions supérieures e.g. 512×512, ceci n'a pas donné de meilleurs résultats au moment de la classification.

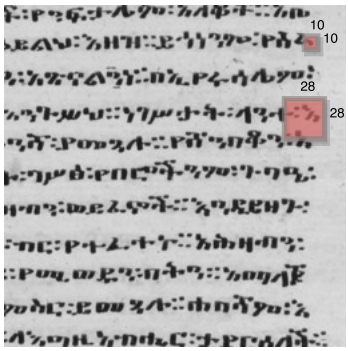


FIGURE 7.14 – Exemple des informations capturer par les filtres, dessinés en rouge, obtenus avec la deuxième, filtre de taille 10×10, et troisième architecture, filtre de taille 28×28.

7.4.1.1 Première architecture

La première architecture correspond à un empilement classique d'auto-encodeurs. La première couche filtre l'image en entrée de $256 \times 256 \times 1$ avec 8 masques de taille $3 \times 3 \times 1$ et une foulée de 1 (*stride* en anglais), c'est-à-dire que le masque est appliqué sans chevauchement de pixels. La couche suivante prend en entrée la sortie de la première couche, qui est un filtre de taille $11 \times 11 \times 8$ et le filtre avec 16 masques d'une taille de $3 \times 3 \times 1$. La troisième couche prend en entrée le filtre de la couche précédente de taille $9 \times 9 \times 16$ et y applique 36 masques de taille $5 \times 5 \times 1$.

Enfin, la dernière couche prend la sortie de la couche précédente, le filtre avec 36 masques et nous obtenons une représentation d'une taille de 36 neurones. Une fois l'image recrée à partir de cette représentation, nous obtenons après calcul de $\hat{x}$ un filtre de taille 13×13. Dans cette première architecture, nous n'utilisons pas de couche de sous-échantillonnage, c'est à dire de couche d'extremum pooler ou max-pooling.

7.4.1.2 Deuxième architecture

La deuxième architecture est similaire à la première, à l'exception de la dernière couche qui est remplacée par un *extremum pooler* qui nous donne un filtre de plus petite taille, égale à 10×10. Avec cette architecture, nous souhaitons étudier l'effet de l'extremum-pooler, mis en dernière couche, sur les résultats de classification.

Les couches d'extremum pooling ne contribuent pas à l'apprentissage du réseau, mais réduisent la taille du problème en obligeant le filtre de convolution à ne prendre que la valeur maximale ou extrême d'un ensemble de neurones. Notons que nous ne pouvons pas exécuter de rétropropagation sur ce type d'architecture. Car nous ne pouvons calculer que la dérivée $\frac{\partial f_n(x_{n-1}, W_n)}{\partial x_n}$ de l'erreur de la sortie de la dernière couche. Un calcul qui nécessite de dériver l'erreur totale obtenue par le réseau et calculée par l'équation 7.21.

$$\frac{\partial E}{\partial x_{n-1}} = \frac{\partial E}{\partial x_n} \times \frac{\partial f_n(x_{n-1}, W_n)}{\partial x_n} \tag{7.21}$$

Où f_n est la fonction de sortie calculée pour la couche n, x_{n-1} est le vecteur donné en entrée par la couche $n-1$ et W_n est la matrice des poids des connexions entre les neurones de la couche n. Enfin, E est l'erreur totale calculée avec la fonction de perte 7.11 à la sortie du réseau.

Ceci est une limite actuelle de notre implémentation framework *N-light-N* que nous utilisons.

7.4.1.3 Troisième architecture

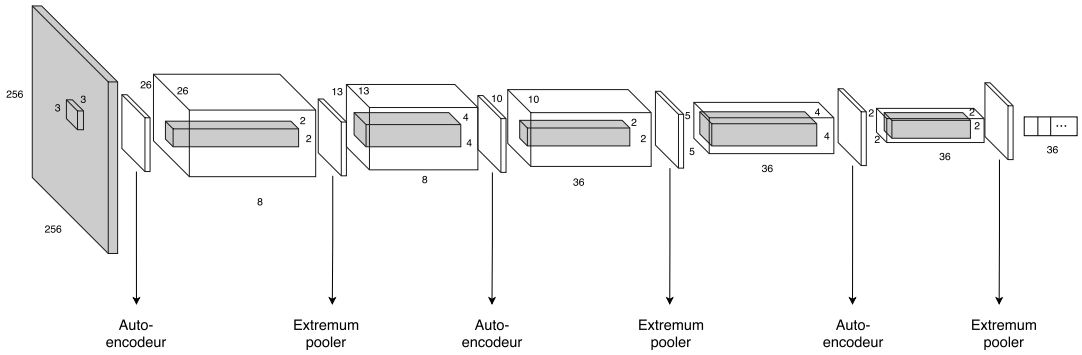
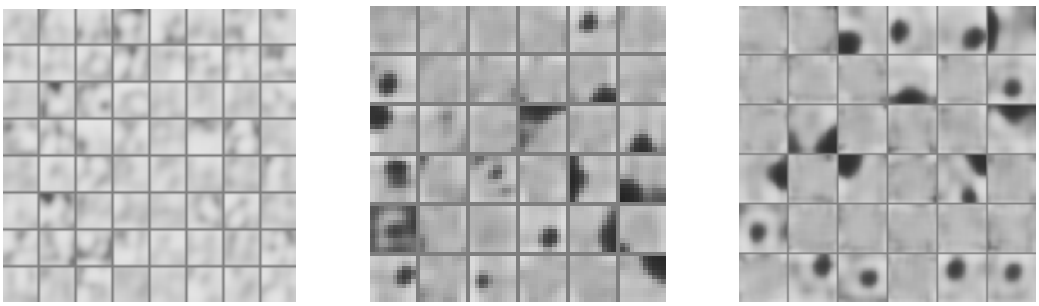


FIGURE 7.15 – Un auto-encodeur avec une seule couche cachée

La troisième architecture correspond à un *sandwich* d'auto-encodeurs empilés avec des couches d'*extremum pooler*, elle est illustrée dans la figure 7.15. Dans la première couche, l'auto-encodeur apprend à partir d'un petit patch 3×3 pixels. La dimension du vecteur d'entrée x est la concaténation des valeurs du signal au niveau de gris des pixels dans le voisinage du patch $3 \times 3 \times 1 = 9$. La foulée est initialisée à 1. Le nombre de neurones ou d'unités cachées est initialisé à 8.

Donc, la première couche de convolution dans l'architecture des auto-encodeurs empilés filtre l'image en entrée de $256 \times 256 \times 1$ avec 8 masques de taille $3 \times 3 \times 1$, avec une foulée (*stride* en anglais) de 1 pixel. On applique à la première couche un *extremum pooler*, qui prend en entrée la sortie de la première couche et la filtre avec 8 masques de taille $2 \times 2 \times 8$. La deuxième couche de convolution prend en entrée la couche sur laquelle nous avons appliqué l'*extremum pooler* et filtre la sortie avec 8 masques. On y applique ensuite un *extremum pooler*. La troisième couche de convolution a 36 filtres de taille $4 \times 4 \times 36$.

Nous obtenons avec la dernière couche une représentation de l'image compressée par 36 neurones. Nous appliquons, par la suite, le processus inverse pour reproduire l'image en entrée. La taille de patch à la sortie du réseau après le calcul de $\hat{x}$ est égale à 28×28.



(*a*) (*b*) (*c*)

FIGURE 7.16 – Exemples des caractéristiques apprises à partir des quatre scriptes (a) première architecture (b) deuxième architecture (c) troisième architecture

Les filtres obtenus avec la dernière couche des trois différentes architectures de SCAE sont illustrés dans la figure 7.16. Nous constatons dans la figure (a) 7.16, que le réseau a été capable d'apprendre des traits courbés qui ne sont pas clairement identifiables comme étant des lettres. La deuxième et la troisième architecture, illustrées respectivement dans la figure (b) 7.16 et (c) 7.16, nous donnent des filtres où nous pouvons distinguer des structures en gouttes, rondes ou plus allongées pour prendre une forme plus ovale. Après une inspection des filtres, nous pouvons passer à la phase d'extraction des caractéristiques.

Pour extraire le vecteur de caractéristiques apprises par l'architecture, que nous avons décrites ci-dessus, nous appliquons aléatoirement 100 fois, pour tous les scripts, et sur chaque page, de chaque manuscrit le patch de taille 28×28. Dans nos expérimentations, pour la phase d'entraînement, ainsi que la phase de test, les pixels

Auto-encodeurs empilés		Precision moy.	Rappel moy.	F1 moy.	Justesse
RF	Ethiopien	**0.59**	0.54	0.56	**55.57%**
	Syriaque	0.52	0.11	0.19	
	Grec	0.49	0.51	0.50	
	Latin	0.58	**0.71**	**0.64**	
SVM	Ethiopien	**0.70**	0.36	0.48	48.91%
	Syriaque	0.57	0.02	0.04	
	Grec	0.41	0.41	0.41	
	Latin	0.48	**0.75**	**0.59**	
ANN	Ethiopien	**0.24**	**0.84**	**0.37**	25.83%
	Syriaque	0.13	0.00	0.01	
	Grec	0.17	0.33	0.19	
	Latin	0.00	0.00	0.00	

TABLE 7.3 – Les résultats des algorithmes de discrimination de scripts entraînée avec les caractéristiques extraites avec l'auto-encodeur convolutif empilé. Les meilleurs résultats sont mis en gras.

dans la bordure de chaque page ne sont pas pris en compte ; seuls les pixels centrés dans un environ de 256×256 sont considérés. La figure 7.9 illustre comment nous appliquons le patch, dessiné en rouge, sur la page manuscrite. La même opération est valable pour la première et deuxième couche.

7.4.2 Classification

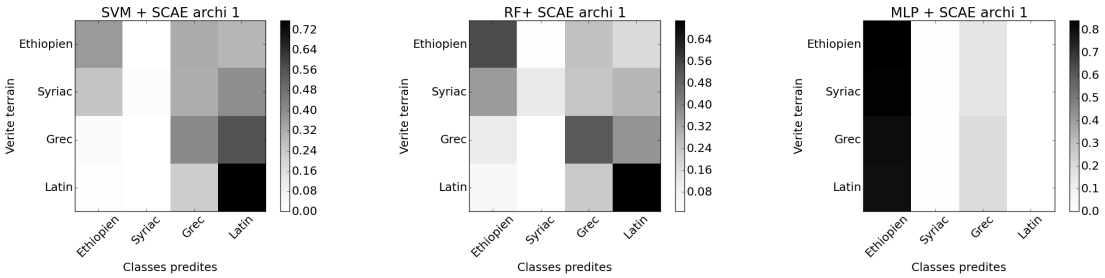
Afin de discriminer les quatre scripts que nous étudions dans ce chapitre, nous avons exploité les caractéristiques apprises par les trois architectures de SCAE pour entraîner les trois algorithmes de notre référentiel empirique à savoir un réseau de neurones artificiels (ANN) à $\frac{\#attributs+\#classes}{2}$ couches, l'algorithme des forêts aléatoires (RF) et l'algorithme des séparateurs à vaste marge (SVM).

7.4.2.1 Résultats

Dans cette partie, nous présentons les résultats d'identifications des scripts : Éthiopien, Syriaque, Grecque et Latin. La présentation des résultats est divisée en deux parties. La première correspond à la discrimination des scripts suite à l'extraction des caractéristiques apprises par les trois architectures SCAE. La deuxième correspond au préapprentissage des deux architectures à convolution présentées précédemment.

7.4.2.2 Première architecture

Dans la table 7.3, nous rapportons les résultats de la première architecture de SCAE qui ne comporte pas de couches d'extremum pooler. Les performances des trois algorithmes, c'est à dire, les forêts aléatoires (RF), les réseaux de neurones artificiels (ANN) et l'algorithme des séparateurs de vastes marges (SVM), ont nettement baissé par rapport à notre référentiel empirique. La détérioration la plus



(a) (b) (c)

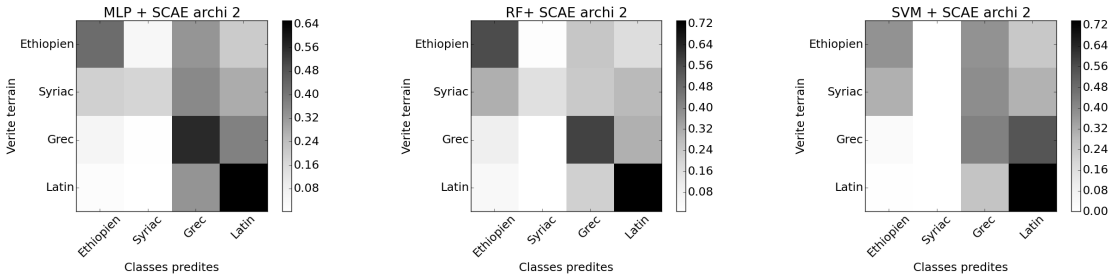
FIGURE 7.17 – Matrice de confusion des algorithmes de classification avec les caractéristiques apprises extraites avec l'architecture 1 de SCAE. La figure (a) illustre les résultats de discrimination du ANN. La figure (b) illustre les résultats de discrimination des forêts aléatoires. Enfin, la figure (b) illustre les résultats de discrimination exécutés par SVM.

Auto-encodeurs empilés		Precision moy.	Rappel moy.	F1 moy.	Justesse
RF	Ethiopien	**0.63**	0.56	0.59	**59.28%**
	Syriaque	0.61	0.16	0.25	
	Grec	0.54	**0.58**	0.56	
	Latin	0.61	0.73	**0.66**	
ANN	Ethiopien	**0.71**	0.42	0.53	52.77%
	Syriaque	0.56	0.17	0.26	
	Grec	0.44	0.56	0.49	
	Latin	0.54	**0.65**	**0.59**	
SVM	Ethiopien	0.68	0.38	0.48	48.77%
	Syriaque	**0.80**	0.00	0.00	
	Grec	0.38	0.42	0.40	
	Latin	0.40	**0.50**	**0.73**	

TABLE 7.4 – Les résultats des algorithmes de discrimination de scripts entraînés avec les caractéristiques extraites par l'auto-encodeur convolutif empilé. Les meilleurs résultats sont mis en gras.

remarquable est celle du ANN qui est passé de 89.07% à 25.83%. Notons également les faibles performances pour la discrimination du script syriaque, et ce pour tous les algorithmes dans la table 7.3.

En observant les matrices de confusion illustrées dans la figure 7.17, nous pouvons constater que les performances de la détection du script syriaque sont faibles comparées aux trois autres scripts. Étonnamment, ANN n'a pas pu détecter une seule instance des pages en Latin donnée lors du test. Nous ne nous pouvons pas, pour l'instant, donner une explication pour les résultats obtenus avec le ANN, surtout en ce qui concerne la discrimination du script Latin, qui comme nous le verrons dans la suite, est toujours bien détecté.



(a) (b) (c)

FIGURE 7.18 – Matrices de confusions des algorithmes de classification avec les caractéristiques apprises extraites avec l'architecture 2 de SCAE. La figure (a) illustre les résultats de discrimination du ANN. La figure (b) illustre les résultats de discrimination des forêts aléatoires. Enfin, la figure (b) illustre les résultats de discrimination exécutés par SVM.

Auto-encodeurs empilés		Precision moy.	Rappel moy.	F1 moy.	Justesse
RF	Ethiopien	0.70	**0.82**	0.75	**71.18%**
	Syriaque	0.69	0.24	0.36	
	Grec	0.68	0.67	0.67	
	Latin	**0.74**	0.78	**0.76**	
ANN	Ethiopien	**0.76**	0.66	**0.71**	64.46%
	Syriaque	0.49	0.34	0.41	
	Grec	0.55	0.65	0.60	
	Latin	0.69	**0.70**	0.69	
SVM	Ethiopien	**0.66**	0.71	**0.68**	56.96%
	Syriaque	0.13	0.00	0.00	
	Grec	0.50	0.43	0.46	
	Latin	0.55	**0.74**	0.63	

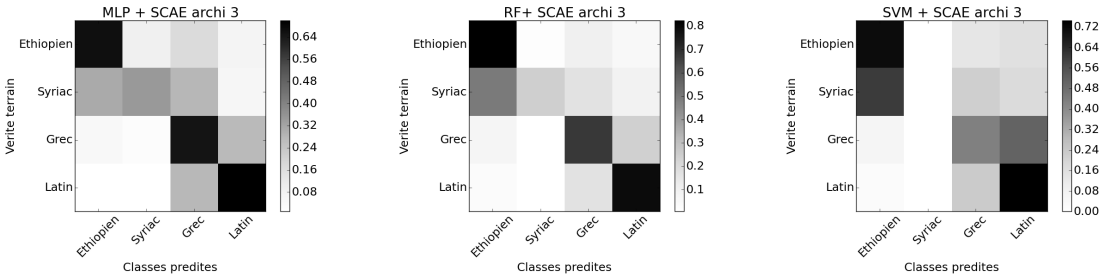
TABLE 7.5 – Les résultats des algorithmes de discrimination de scripts entraînée avec les caractéristiques extraites avec l'auto-encodeur convolutif empilé. Les meilleurs résultats sont mis en gras.

7.4.2.3 Deuxième architecture

Dans la table 7.4, nous rapportons les résultats des trois algorithmes : RF, ANN et SVM. Ces derniers sont entraînés avec les caractéristiques apprises par la deuxième architecture SCAE que nous avons présentées dans la section 7.4.1.1. Les performances globales des algorithmes se sont améliorées par rapport à l'architecture précédente. Les forêts aléatoires ont pris les devants sur les réseaux de neurones artificiels (ANN) avec une justesse de 59.28% en comparaison avec le référentiel construit avec les caractéristiques manuelles. Notons les performances très faibles pour l'identification du script syriaque, qui rappelons-le, a le moins de données dans notre *vérité terrain*.

En analysant les matrices de confusion dans la figure 7.19, nous constatons que les ANNs ont une meilleure capacité de généralisation pour l'Éthiopien, le Grecque

et le Latin. Malgré un score de justesse plus élevé, les forêts aléatoires ont plus de mal que les ANNs à identifier le script syriaque. L'algorithme SVM fournit les performances les plus faibles, et n'arrive même pas à identifier une seule instance du script syriaque.



(a) (b) (c)

FIGURE 7.19 – Matrice de confusion des algorithmes de classification avec les caractéristiques apprises extraites avec l'architecture 3 de SCAE. La figure (a) illustre les résultats de discrimination du ANN. La figure (b) illustre les résultats de discrimination des forêts aléatoires. Enfin, la figure (b) illustre les résultats de discrimination exécutés par SVM.

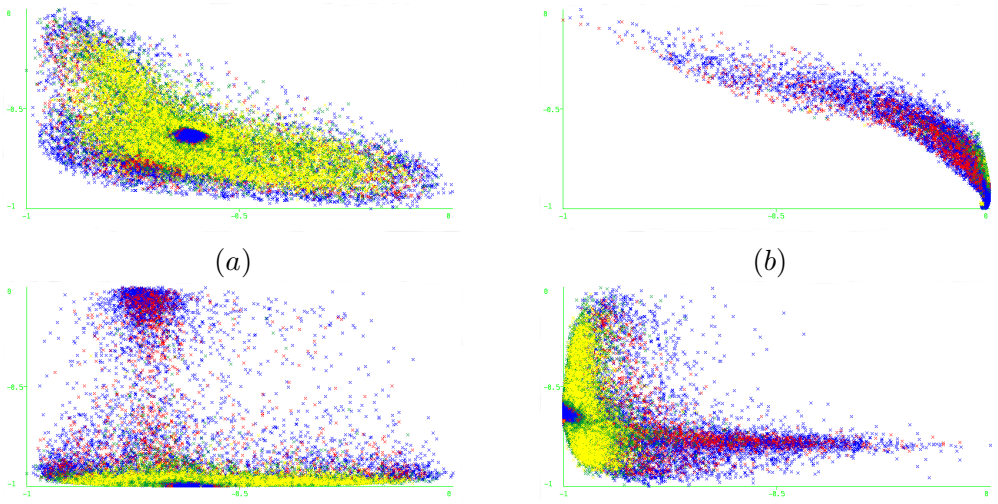
7.4.2.4 Troisième architecture

Dans la table 7.5, nous présentons les résultats de la discrimination des scripts après extraction des caractéristiques apprises avec la troisième architecture de SCAE. Contrairement au référentiel construit avec les caractéristiques manuelles, les forêts aléatoires obtiennent de meilleurs résultats que les réseaux de neurones artificiels, avec une justesse égale à 71.18%. Les résultats des trois algorithmes se sont nettement améliorés par rapport aux deux premières architectures de SCAE, mais n'ont pas atteint les performances des algorithmes entraînés avec les caractéristiques HOG. Nous pouvons clairement observer à travers la figure 7.19, la pauvre performance pour la discrimination du script syriaque, surtout pour le SVM qui, comparé à ces résultats, possédait une bonne capacité de généralisation quand il a été entraîné avec les caractéristiques manuelles de HOG.

Dans cette partie, nous avons présenté les résultats obtenus avec différentes stratégies de classification. Le point commun entre ces stratégies ce sont les caractéristiques apprises avec les trois différentes architectures de SCAE que nous avons conçues dans la section 7.4.1. Après plusieurs exécutions, en optimisant les différents paramètres des classifieurs (e.g. nombre d'arbres dans les forêts aléatoires ou nombre de neurones dans les réseaux de neurones artificiels), nous avons constaté qu'il n'y avait pas de changement significatif dans les résultats. Nous avons donc conclu que les caractéristiques apprises étaient seules responsables des résultats obtenus et nous avons décidé, après plusieurs optimisations des trois architectures, de visualiser et par la suite analyser ces caractéristiques.

7.4.3 Analyse des résultats

À partir d'un ensemble réduit de données sans un prétraitement préalable (e.g. nettoyage du bruit dans l'image, binarisation, détection des mots, détection des composantes connexes), nous espérions qu'un pré entraînement non supervisé, couche après couche, nous permettrait d'extraire des caractéristiques de plus en plus abstraites et donc, par la suite, pertinentes pour une classification selon un critère supervisé. Avec la série d'expériences que nous avons menées, nous voulions tester notre hypothèse qui stipule qu'un empilement d'auto-encodeurs nous permettra d'obtenir ce genre de caractéristiques.



(*a*) (*b*) (*c*) (*d*)

FIGURE 7.20 – Visualisation des caractéristiques apprises par l'auto-encodeur. Chaque point correspond à une valeur dans le vecteur de caractéristiques apprises après la phase d'encodage dans les différentes architectures. La couleur bleue correspond au script Éthiopien, la couleur rouge au Syriaque, la couleur verte au Grec et enfin le jaune au Latin.

Nous nous attendions, donc, à ce que les représentations apprises par les trois architectures SCAE nous donnent de meilleurs résultats que ceux présentés dans notre référentiel empirique construit avec l'extraction de caractéristiques manuelles. Cependant, la meilleure performance obtenue est celle des forêts aléatoires (RF), entraînées avec les caractéristiques extraites de la troisième architecture SCAE. Les RF ont obtenu un taux de justesse égale à 71.18% (voir table 7.5) comparé au 89.07% (voir table 7.2), obtenu avec un réseau de neurones artificiel ou ANN, entraînés avec les caractéristiques manuelles HOG.

En observant les caractéristiques de la troisième architecture dans la figure 7.16 (b), nous pouvons distinguer des structures en gouttes laissant à présager que ces représentations étaient utiles pour le RF. Malgré que les caractéristiques illustrées dans la figure 7.16 (a), apprises par la première architecture semblent plus intéressantes. Car, nous pouvons distinguer des contours de lettres et des courbatures de

traits, laissant à penser que nous pouvons avoir de meilleurs résultats de discrimination. Ceci n'a pas empêché d'avoir le plus mauvais taux de justesse avec 25.83% obtenu avec un ANN entraîné sur ces caractéristiques.

Suite au mauvais résultat que nous avons obtenu avec la première architecture, nous avons eu l'intuition suivante : comme nous travaillons au niveau des pixels d'une page manuscrite, le risque que l'architecture SCAE apprenne une représentation inintéressante est plus fort comparé à un SCAE entraîné au niveau d'un caractère ou un mot ou une composante connexe. En effet, le patch de l'architecture SCAE a plus de chance de tomber sur du blanc ou du bruit dans l'arrière-plan que sur des lettres manuscrites.

Cette intuition revient, donc, à attaquer le problème à la phase de l'encodage afin que le SCAE puisse apprendre une représentation des images des documents données en entrée. C'est pour cette raison que nous avons décidé d'introduire, dans la deuxième et troisième architecture SCAE, de la dispersion dans les valeurs des neurones en ajoutant des couches qui orientent le gradient vers le neurone avec la valeur maximale.

Les caractéristiques extraites avec les auto-encodeurs à convolution sont visualisées dans la figure 7.20, les classes correspondant aux scripts Éthiopien, syriaque, Grec et Latin sont colorées respectivement en bleu, rouge, vert et jaune. Nous pouvons clairement observer que dans les figures 7.20 (a), 7.20 (c) et 7.20 (d) les caractéristiques correspondant au script latin sont clairement prédominantes, les autres scripts y sont juste interposés.

Nous observons dans la figure (c) 7.20 que les deux scripts Éthiopien et syriaque se détachent un peu, mais sans pour autant constituer un nuage complètement indépendant des scripts grec et latin. Ces derniers sont toujours interposés. Selon la visualisation, nous pouvons conclure que le SCAE confond entre le latin et le grec, le Syriaque et l'Éthopien. Ceci n'est pas vraiment étonnant vu la manière dont ils sont écrits avec un large trait (voir figure 7.1). Il nous faut, donc, plus de données pour que le CAE généralise son apprentissage et distingue des caractéristiques plus fines et qu'il soit plus sensible à la variation.

Discussion

Les résultats de visualisation montrés dans la figure 7.20 viennent appuyer ceux démontrés dans les tables 7.3, 7.4, 7.5. La capacité de généraliser l'apprentissage d'une représentation à caractère non supervisé apprise à partir d'une quantité limitée de données et qui sont, de plus, non traitées est mise en doute. Il est vrai que l'ajout des couches d'extremum pooling a aidé à améliorer considérablement la justesse des résultats en les faisant passer de 55.57% à 71.18% pour les forêts aléatoires. Cependant, ceci n'égale pas les performances de notre référentiel empirique avec un taux de justesse égale à 86.15% pour le même algorithme (i.e. forêt aléatoire).

Nous ne pouvons pas avoir une seule explication à ces résultats, vu l'espace de paramètres que nous devons explorer. Par exemple, comme nous le savons, les auto-encodeurs construisent une représentation compacte d'une image, cette repré-

sentation doit permettre une reconstruction optimale de l'image d'origine, ce qui revient à optimiser la dimension des caractéristiques utilisées lors de la reconstruction. Cependant, nous sommes encore incertains si la dimension optimisée lors de l'apprentissage de l'auto-encodeur est optimale en entrée d'un algorithme de discrimination ou classification.

Aussi, dans la procédure d'entraînement, nous prenons en compte un nombre de patchs aléatoires à partir de l'image. Ces patchs peuvent être tout blancs, ce qui rend la généralisation de la sortie du masque au moment de la convolution encore plus difficile. On est toujours aussi exposé au risque de perte du signal d'origine à cause d'une mauvaise initialisation des poids au sein de la toute première couche des architectures. Une possible solution à entreprendre consiste en l'utilisation du *finetuning*.

L'ajustement fin ou *finetuning* en anglais est l'approximation de la meilleure valeur possible des paramètres du réseau que nous avons déjà entraîné et est basé sur le principe de l'apprentissage par transfert [Pan 2010]. Dans le cas des SCAE, cette technique peut augmenter considérablement les performances du réseau où nous traitons toutes les couches comme un seul modèle. Nous n'avons pas utilisé l'ajustement fin dans nos expérimentations, car, et comme expliquée précédemment, nous ne pouvons pas utiliser de rétropropagation sur les couches avec un gradient orienté vers le neurone avec la valeur maximale. Ceci est pour l'instant une limitation du framework *N-light-N*.

Notons que nous avons, aussi, essayé d'optimiser nos architectures en explorant les auto-encodeurs débruitant [Vincent 2008] où il s'agit d'apprendre une représentation utile à la corruption synthétique de l'entrée. L'auto-encodeur débruitant ne diffère pas de l'auto-encodeur ordinaire, à part le fait que les données x en entrée sont corrompues, dénotées $\tilde{x}$, et que les données en sortie sont aussi données pour effectuer la reconstruction $z = g_{\theta'}(f_\theta(\tilde{x}))$ de la représentation cachée $y = f_\theta(\tilde{x})$. Malheureusement, la nature de nos données ne nous permet pas d'appliquer ce genre de technique qui a besoin de données nettoyées et normalisées.

Notons, aussi, que nous avons changé la résolution de l'image en entrée en passant de 256×256 à 512×512, afin de vérifier si une augmentation dans le nombre de pixels aurait un impact sur le taux de justesse obtenu. Nous n'avons pas eu de changement sensible qui mériterait un intérêt particulier.

7.5 Contribution pour l'apprentissage supervisé de scripts avec les architectures à convolution

Dans la section précédente, nous nous sommes intéressés à l'apprentissage non supervisé des caractéristiques que nous avons utilisées par la suite pour entraîner les trois algorithmes utilisés pour construire notre référentiel empirique. Nous avons également analysé les résultats de discrimination obtenus avec le rajout de deux couches connexes de neurones pré entraînés.

Dans cette section, nous explorons une architecture plus simple faite d'un petit

nombre de couches convolutives qui sont normalisées par des couches de *max pooling* et donc appliquées sur un réseau de type *Sparse*. Nous analysons d'abord le rôle de la fonction d'initialisation des poids des connexions neuronales. Ensuite, l'optimisation des différents paramètres de l'algorithme de descente de gradient. Nous voulons jauger, à travers les expérimentations, l'effet de l'optimisation des paramètres sur la justesse des résultats de la discrimination des quatre scripts étudiés.

La motivation d'entraîner des couches profondes vient des travaux de Y. Bengio, qui stipule que si nous avons beaucoup de poids dans notre réseau, alors les minimums locaux deviennent, à peu près, égaux aux minimums globaux. Autrement dit, lorsque nous exécutons une descente de gradient, la probabilité de la direction de la descente est proportionnelle au nombre de poids. Néanmoins, dans les travaux sur les architectures profondes [Hinton 2006b, Bengio 2006, Erhan 2009, Larochelle 2009] initiés après l'année 2005, les chercheurs ont supposé que c'était difficile d'entraîner un réseau constitué de plusieurs couches avec l'algorithme de descente de gradient, ceci à cause des mauvais résultats obtenus. C'est, entre autres, avec les travaux de [Krizhevsky 2012] que les architectures profondes ont été remises dans la course.

Les auteurs de [Krizhevsky 2012] ont entraîné un large réseau à convolution avec 60 millions de paramètres et un jeu de données se composant de plus d'un million d'images annotées avec 1000 classes. Les bons résultats obtenus ont prouvé que c'était possible d'entraîner des architectures profondes à condition d'apporter une attention particulière à l'optimisation des paramètres de la descente de gradient, mais aussi à l'initialisation aléatoire des poids des connexions neuronales.

Notons, aussi, que les auteurs ont utilisé deux améliorations majeures pour les architectures profondes : la transformation non linéaire appelée *Rectified Linear Unit* (ReLU) [Nair 2010] pour améliorer la convergence de l'apprentissage lors de l'activation des neurones dans le réseau, et la technique du *Dropout* [Srivastava 2014] pour réduire le sur apprentissage. Nous n'utiliserons pas de *Dropout* dans nos expérimentations, mais la méthode de dégradation des pondérations appelée *weight decay*. Cette méthode consiste à rajouter une pénalité à la fonction d'erreur. Une étude comparative des deux méthodes de régulation de l'apprentissage (i.e. le *Dropout* et le *weight decay*) a été présentée dans les travaux de [Shi 2016]. Notons que nous remplacerons également, dans ce qui suit, la fonction d'activation *softsign* par la fonction ReLU.

Dans le cadre les auteurs de [Bluche 2013] ont utilisé les réseaux de neurones à convolution afin d'extraire les caractéristiques pour la reconnaissance des mots écrits à la main. Dans leur article, les auteurs proposent une comparaison entre les caractéristiques extraites manuellement et celles apprises de manière supervisée. Pour la tâche de la classification, les auteurs ont combiné les réseaux de neurones à convolution et les chaînes de Markov cachées. Leurs résultats prouvent que la combinaison avec les architectures profondes permet d'obtenir de meilleurs résultats que l'utilisation seule d'une chaîne de Markov cachée. Également, les auteurs de ce travail [Du 2015] ont utilisé les réseaux de neurones à convolution pour apprendre de manière supervisée les caractéristiques au niveau des caractères chinois. Ils ont par la suite, donné en entrée les caractéristiques apprises à un algorithme de classification

par prototypage des caractéristiques.

Nous utiliserons, dans cette partie, la plateforme *Caffe* pour la conception de nos architectures profondes convolutives. *Caffe* est développé par l'équipe de [Jia 2014] à Berkley et a connu, auprès de la communauté travaillant sur les architectures profondes, un franc succès. Notons qu'il y a d'autres plateformes aussi efficaces que *Caffe*, cet article [Bahrampour 2015] les compare au niveau de la vitesse, de la consommation de ressources, etc. Notons aussi que *Caffe* offre la possibilité d'implémenter des réseaux *Sparse* c'est-à-dire que la valeur des neurones en sortie est dans l'intervalle $[0, 1]$.

7.5.1 Sur l'importance de l'initialisation des paramètres

L'implication majeure des travaux de [Krizhevsky 2012], à notre sens, est la mise en évidence de l'importance de l'initialisation des paramètres (i.e. poids des connexions neuronales, paramètres de la descente de gradient) au début de chaque entraînement. Ceci a été souligné par les travaux de [Sutskever 2013b, Sutskever 2013a] sur les réseaux de neurones récurrents. En effet, la méthode d'entraînement d'une structure neuronale, qu'elle soit profonde ou pas consiste en la transformation de la problématique d'apprentissage supervisée en une problématique d'optimisation des paramètres $\theta = \{W, V, b, c\}$.

L'optimisation est traduite par une approximation des valeurs qui minimisent la fonction objectif ou de perte $f(\theta)$ (un exemple de $f(\theta)$ est illustré dans la figure 7.21). L'approximation est exécutée par l'algorithme de descente de gradient et par une rétropropagation sur les poids des connexions neuronales. Cependant, cette optimisation qui est basée sur une initialisation aléatoire des paramètres en amont résulte en de mauvaises performances de généralisation et même de mauvaises solutions au niveau des données d'entraînement [Larochelle 2009, Erhan 2009, Hinton 2006a].

Ceci s'explique par le choix d'un mauvais bassin d'attraction dans l'espace des paramètres du réseau de neurones, résultant en l'exploration de mauvais minimaux locaux, plateaux, mais aussi des points col (*saddles point* en anglais), comme l'avaient démontré les auteurs de ce travail [Dauphin 2014]. Néanmoins, même avec le choix du bon bassin d'attraction, nous ne pouvons pas être sûrs que le modèle aura une bonne généralisation, comme c'était le cas avec notre première architecture de SCAE étudiée dans la section 7.4.1.

Face à une mauvaise généralisation, l'on parle de sur apprentissage qui nécessite l'usage de la régularisation pour le choix du modèle le plus simple. Nous nous retrouvons dans la problématique de la sélection du modèle le plus simple ou de l'hypothèse la plus courte (Définition A.9) qui aura le plus de capacité à généraliser ses prédictions en vue de nouveaux exemples.

Avec les travaux de [Hinton 2006a], une nouvelle stratégie de régularisation basée sur un apprentissage non supervisé a vu le jour. Il s'agit d'initialiser les paramètres (poids et biais) avec un critère non supervisé avant d'entamer l'apprentissage supervisé avec la descente de gradient. Les auteurs ont utilisé une architecture d'empilement de machines de Boltzmann restreintes (RBM pour *Restricted Boltzmann*

Machines).

D'autres travaux ont exploré l'empilement d'auto-encodeurs ordinaires [Bengio 2007b] (SAE pour *Stacked Autoencoders*, ou encore l'empilement d'auto-encodeurs débruiteurs [Vincent 2010] (SDAE pour *Stacked Denoising Autoencoders*). Dans nos travaux présentés dans la section 7.4, nous avons utilisé un empilement d'auto-encodeurs de type convolutif (SCAE). Le rajout de plusieurs couches d'auto-encodeurs peut être une manière d'éviter l'initialisation aléatoire des poids du réseau de neurones dédié à la classification, cependant il faut toujours que l'on choisisse les poids du réseau de préapprentissage que ce soit un RBM, un SDAE, ou un SCAE.

Rappelons que la convolution du signal en entrée, dans ce type d'architecture, est aussi dépendante des valeurs initiales que nous attribuons aux poids des connexions neuronales. Si les poids des connexions entre les neurones du réseau sont trop petits, alors le signal d'origine rétrécit avec chaque multiplication et transformation linéaire (i.e. une opération de convolution), jusqu'à ce qu'il soit dégradé, atteins une valeur nulle (*decay to zero*), et ne soit plus utilisable en sortie du réseau. Par contre si les poids dans le réseau sont trop grands, alors le signal d'origine croît lors de la convolution, jusqu'à ce qu'il devient trop grand pour être utile. Ceci a une conséquence directe sur l'entraînement qui ne convergera jamais vers une valeur et le système peut devenir par la suite chaotique.

Initialisation des poids

Nous avons utilisé dans nos travaux sur les SCAE une initialisation aléatoire des poids des connexions neuronales qui est tirée d'une distribution uniforme où $W \in]-\frac{1}{\sqrt{n_{in}}};\frac{1}{\sqrt{n_{in}}}[$. Cet intervalle a été déterminé empiriquement, où nous avons constaté après plusieurs essais que l'utilisation de la racine nous évitait de perdre le signal en entrée n_{in}.

Nous utiliserons dans cette partie une autre méthode pour l'initialisation des poids des connexions neuronales. La valeur initiale des poids est tirée d'une distribution gaussienne, avec une moyenne nulle et une variance décrite par l'équation suivante

$$Var(G) = \frac{1}{n_{in}} \tag{7.22}$$

Où G est la distribution gaussienne, et n_{in} est le nombre de neurones donné en entrée à la distribution. Afin de stabiliser le signal en entrée du réseau (i.e. éviter qu'il explose ou diminue jusqu'à disparaître complètement), notre objectif consiste à garer la même variance du signal lors de l'exécution de l'algorithme de descente de gradient. L'on appelle ce type d'initialisation aléatoire, l'initialisation de Xavier [Glorot 2010].

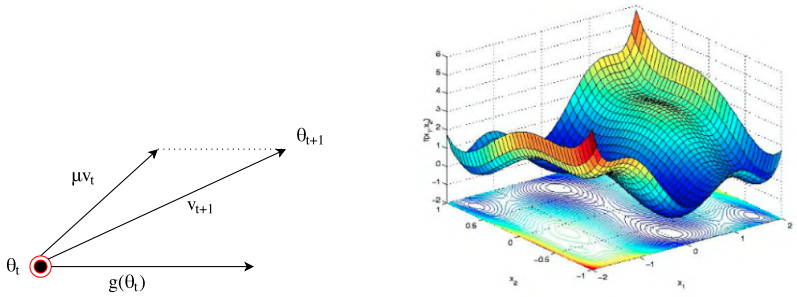
La variance de la distribution gaussienne telle qu'elle est calculée dans l'équation 7.22, est une implémentation de la plateforme d'apprentissage des architectures profondes appelée *Caffe* [Jia 2014], que nous utilisons dans cette partie afin de mener

nos expérimentations. La variance telle qu'elle est calculée dans les travaux d'origine de [Glorot 2010] est donnée par l'équation 7.23.

$$Var(G) = \frac{2}{n_{in} + n_{out}} \tag{7.23}$$

Où n_{out} est le nombre de neurones en sortie. Afin de préserver la variance du signal en entrée ainsi qu'en sortie après l'application de l'algorithme de rétropropagation, la variance est calculée sur la moyenne des neurones donnés en entrée et en sortie.

Élan du gradient



(a) (b)

FIGURE 7.21 – (a) Calcul de la direction du gradient à l'étape θ_{t+1}, Où $g(\theta_t) = \varepsilon \nabla f(\theta_t)$. (b) Illustration d'une fonction objectif $f(\theta)$.

Étant donné la fonction objectif $f(\theta)$ que nous souhaitons minimiser, l'élan μ est donné par :

$$v_{t+1} = \mu v_t - \varepsilon \nabla f(\theta_t) \tag{7.24}$$
$$\theta_{t+1} = \theta_t + v_{t+1} \tag{7.25}$$

Où $\varepsilon > 0$ est le taux d'apprentissage, $\mu \in [0, 1]$ est le coefficient de l'élan et $\nabla f(\theta_t)$ est le gradient à θ_t. Nous avons appelé le coefficient μ élan, par abus de langage et suite à la traduction du terme anglais *momentum* que nous retrouvons dans l'état de l'art. Cependant, μ représente en réalité le frottement exercé sur la vitesse v, il a pour effet la réduction et l'alourdissement de l'énergie cinétique du système.

Si nous considérons l'espace de la fonction objectif $f(\theta)$, illustrée dans la figure (b) [9] 7.21, comme étant un massif rocheux, que nous nous positionnons en haut d'une colline et que notre but est d'atteindre le bas de cette colline, alors le frottement μ que nous exerçons sur la vitesse de déplacement nous "aide" à nous arrêter, une

9. Figure prise des slides Multilayer Neural Networks par Leon Bottou présenté dans *Deep learning summer school, Montreal, 2015*

fois la descente terminée. La direction du gradient à θ_{t+1} est calculée par l'équation 7.25 qui est illustrée dans la figure (a) 7.21.

Pour l'optimisation de la fonction objectif, nous nous sommes concentrés sur les paramètres suivants :

— Le taux d'apprentissage ε.
— L'élan de l'apprentissage μ. Notons que nous nous intéressons uniquement à l'élan classique de l'apprentissage. Nous n'explorons pas l'effet du paramètre de Nestrov pour l'accélération du gradient (NAG), bien que dans leur article [Sutskever 2013b] les auteurs démontrent que le NAG est sensiblement meilleur que le paramètre de l'élan de l'apprentissage classique.
— La fonction de l'initialisation aléatoire des poids des connexions des neurones.
— La profondeur du réseau.

7.5.2 Classification

Dans cette partie, nous avons fait varier le taux d'apprentissage, l'élan de l'apprentissage, changé la fonction d'initialisation aléatoire des poids et proposé une nouvelle architecture convolutive illustrée dans la figure 7.22. Nous rapportons également, les résultats de l'entraînement avec nos données de l'architecture proposée par [Krizhevsky 2012].

Les deux architectures utilisent la fonction d'activation *ReLU* (*Rectified Linear Unit*) où la sortie de chaque neurone est calculée avec la fonction $f(x) = \max(0, x)$. En comparaison avec la fonction $f(x) = tanh(x)$ par exemple, l'utilisation de ReLU ne nous confronte pas au problème de la perte du signal suite à l'application de convolutions successives. De plus, et comme a démontré [Krizhevsky 2012], ReLU est 6 fois plus rapide en temps d'entraînement que les autres fonctions d'activations présentées dans la sous-section 7.2.1, ceci est essentiellement dû à la suppression de la normalisation ainsi que l'absence de tout calcul exponentiel.

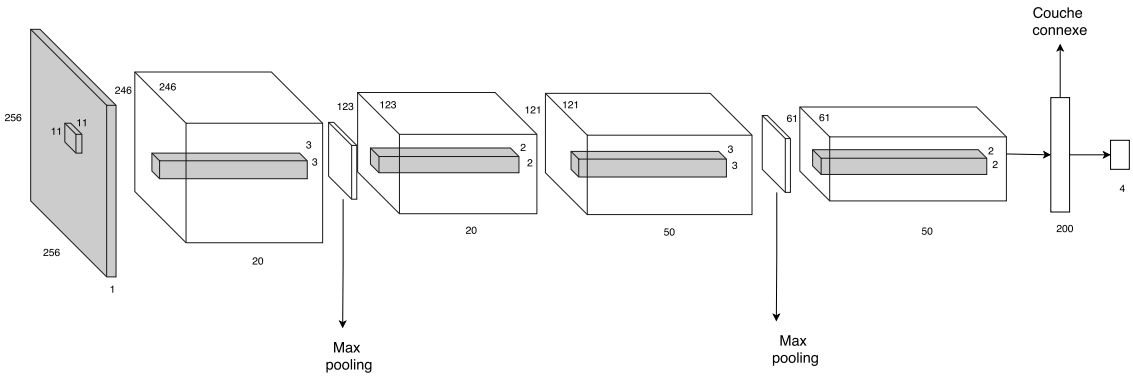


FIGURE 7.22 – Architecture d'un réseau convolutif avec des couches de *max pooling*

Dans la figure 7.22, l'architecture du réseau est constituée de quatre couches

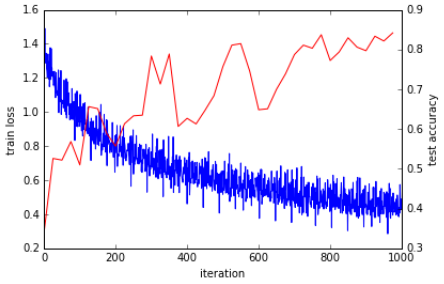


FIGURE 7.23 – illustration de la sortie de la fonction de perte lors de l'entraînement et de la justesse des résultats calculés sur les données de test

de convolution, 2 couches de *maxpooling* et deux couches de neurones connexes [10]. La sortie de la première couche connexe composée de 200 neurones est donnée en entrée à la deuxième couche connexe. Une fonction *softmax* calcule alors un ordre de probabilités afin de classifier le signal sur lequel nous avons appliqué une convolution avec l'un des quatre labels correspondants à l'Éthiopien, le Syriaque, le Grec ou le Latin.

La première couche convolutive filtre l'image en entrée de $256 \times 256 \times 1$ avec 20 masques d'une taille de $11 \times 11 \times 1$ et une foulée d'un pixel. La couche convolutive suivante prend en entrée la sortie de la première couche et y applique un filtre avec 50 masques d'une taille de $3 \times 3 \times 20$. La deuxième et quatrième couche, elles correspondent à des couches de *maxpooling*. La deuxième couche exécute 20 masques d'une taille de $2 \times 2 \times 20$ et une foulée de 2 pixels. La quatrième couche exécute 50 masques d'une taille de $2 \times 2 \times 50$ et une foulée de 2 pixels.

Notons que nous avons également essayé d'autres configurations avec par exemple plus de couches convolutives, un maxpooling à la sortie de chaque convolution ou aussi un seul maxpooling. À la fin de toutes les convolutions. Nous n'avons pas eu de meilleurs résultats par rapport à l'architecture présentée au-dessus.

	Taux ε	**Élan μ**	**weight decay**	**justesse**
Configuration # 1	10^{-4}	0.7	4×10^{-4}	**85%**
Configuration # 2	10^{-4}	0.6	4×10^{-4}	76%
Configuration # 3	10^{-5}	0.7	4×10^{-4}	43%

TABLE 7.6 – Les différentes initialisations de paramètres, pour la même architecture.

Nous rapportons dans le tableau 7.6 trois différentes configurations de l'architecture illustrée dans la figure 7.22. La configuration avec un taux de 85% de justesse est composée d'un taux d'apprentissage $\varepsilon = 10^{-4}$, un élan $\mu = 0.7$ et un *weight decay* de 4×10^{-4}. Ces résultats sont comparables en performance à ceux rapportés dans le référentiel empirique (voir tableau 7.2) et plus précisément les 89% de justesse obtenue par le réseau de neurones (ANN) entraîné avec des caractéristiques HOG.

10. Dans une couche connexe, tous les neurones sont connectés entre eux au sein d'une même couche et avec ceux de la couche suivante.

Dans la figure 7.23, nous avons illustré deux courbes. La première en rouge (ascendante) est celle du taux de justesse à travers les itérations. La deuxième en bleu (descendante) est la valeur de la fonction objectif que nous souhaitons minimiser, appelée aussi perte (en anglais *train loss*). La plus petite valeur atteinte est de 0.27, la valeur moyenne de perte est de 0.41. Nous avons atteint une justesse de prédiction sur les données de test d'une valeur de 85% et ce après 1000 itérations. Quant au comportement "bruité" de la courbe de la justesse sur les données de test, il est expliqué par l'utilisation du *weight decay* pour éviter le sur apprentissage.

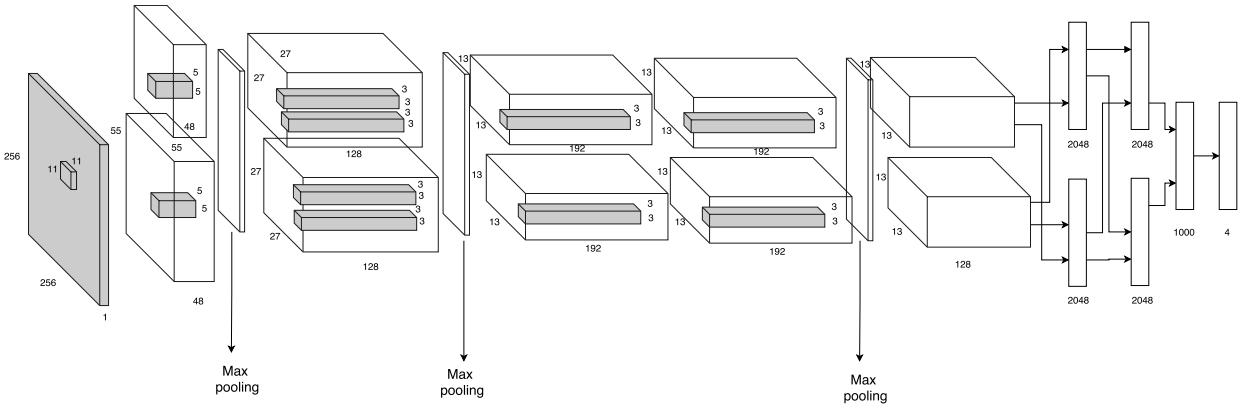


FIGURE 7.24 – Architecture d'un réseau convolutif avec des couches de *max pooling*

Dans la figure 7.24, l'architecture du réseau de [Krizhevsky 2012] appelé Alexnet est constituée de cinq couches de convolution, trois d'entre elles sont suivies par des couches de *maxpooling*, à la fin des trois couches de neurones connexes nous avons connecté une quatrième avec quatre neurones dont la distribution est calculée avec un *softmax*.

La première couche convolutive filtre l'image en entrée de $224 \times 224 \times 1$ avec 96 masques d'une taille de $11 \times 11 \times 1$ et une foulée de 4 pixels. La seconde couche convolutive prend la sortie de la couche de *maxpooling* et filtre cette dernière avec 256 masques d'une taille de $5 \times 5 \times 48$. La troisième couche prend la sortie de la deuxième couche après normalisation et application d'un *maxpooling* sur les neurones, elle y applique 384 masques d'une taille de $1 \times 1 \times 256$. La quatrième applique sur la sortie de la troisième 384 masques d'une taille de $1 \times 1 \times 192$. Quant à la cinquième, elle possède 256 masques avec une taille de $1 \times 1 \times 192$. Les couches connexes possèdent chacune 4096 neurones.

Alexnet a été originellement construit pour la classification de 1.2 million d'images en 1000 catégories. Le réseau a été entraîné sur les trois canaux RGB (i.e. sur des images en couleur). Le signal en entrée est tridimensionnel. Comme nous l'avons illustré dans la figure 7.1, nous travaillons avec des images au niveau du gris. Nous n'avons donc pas pu utiliser un réseau déjà entraîné et avons dû en entraîner un nouveau en précisant qu'il ne devait prendre en considération qu'un seul canal, celui du gris.

La fonction d'activation non linéaire ReLU est utilisée après chaque couche de convolution ou couche connexe. Pour la régularisation de l'apprentissage, le *Dropout* est utilisé avec une valeur de 0.5 et ce dans les deux couches connexes . Nous n'avons pas appliqué de *Dropout* à la couche des quatre neurones connexes que nous avons ajoutés à la fin de l'architecture. Nous avons gardé un taux d'apprentissage $\varepsilon = 10^{-4}$ et un élan $\mu = 0.7$. Nous avons obtenu avec cette configuration un taux de justesse égale à 82%.

7.5.3 Analyse des résultats

Nous constatons que les deux architectures convolutives ont obtenu de meilleurs résultats que les auto-encodeurs. Nous pensons que ceci est principalement dû aux caractéristiques extraites. Face à des données réduites comportant une quantité non négligeable de bruits (voir figure 7.2), il semblerait d'après nos résultats empiriques qu'une architecture convolutive simple soit plus appropriée qu'une architecture de type auto-encodeurs.

Les résultats empiriques que nous avons obtenus nous laissent penser que les auto-encodeurs nécessiteraient plus de données afin de construire une meilleure représentation de toutes les caractéristiques que nous pouvons trouver dans un manuscrit ancien. Le compromis entre le choix du modèle d'apprentissage et le rajout de plus de données lors de l'entraînement de ce dernier, a été traité dans les travaux de [Zhu 2012].

Notons également que l'architecture que nous avons proposée, et qui est illustrée dans la figure 7.22, a obtenu en moyenne un résultat légèrement meilleur (une différence de 3% dans le taux de justesse) que celui d'Alexnet. Cependant, nous pensons que l'architecture Alexnet est capable de nous fournir un meilleur résultat si nous avons eu la possibilité de récupérer une version entraînée sur plusieurs images de documents au niveau du gris. En effet, en appliquant un apprentissage par transfert plusieurs travaux [Shin 2016] rapportent de meilleurs résultats par rapport à un apprentissage à partir de zéro.

Les résultats obtenus démontrent également l'importance des différents paramètres que nous avons utilisés lors de l'apprentissage. Comme nous avons observé dans le tableau 7.6, un changement sensible dans la valeur de certains paramètres peut résulter dans une chute importante dans le taux de justesse. Par exemple, passant de 85% à 43% lors d'une modification du taux d'apprentissage ε. Ou encore de 85% à 76% lors d'une modification de l'élan μ.

Notons finalement l'importance du schéma de l'initialisation aléatoire des poids des connexions neuronales. Dans la partie des SCAE, nous avons utilisé une fonction d'initialisation que nous avons tirée d'un ensemble de tests empiriques. Dans cette partie, nous avons utilisé une fonction d'initialisation avec une distribution gaussienne où nous avons évité de perdre le signal en réduisant la variance. Il semblerait que cette dernière soit plus efficace toujours d'après nos résultats empiriques.

Dans cette partie, nous nous sommes intéressés au problème de l'identification des scripts dans des manuscrits comportant l'Éthiopien, le Syriaque, le Grec et le Latin. Nous avons approché le problème en menant une étude empirique où nous avons adopté une analyse en termes des pixels au niveau d'une page entière issue d'un manuscrit donné. Cette analyse est basée sur l'apprentissage d'un ensemble de caractéristiques par l'application d'une suite de transformations non linéaires sur les pixels de la page donnée. Cette suite de transformations, appelée convolutions, est appliquée sur ce signal (i.e. ensemble de pixels) à travers une couche de neurones en entrée, une ou plusieurs couches cachées et une couche de sortie. Ces couches forment ce que l'on appelle une architecture profonde.

Deux catégories d'architectures ont été étudiées empiriquement dans ce chapitre. La première appelée autoencodeur, nous fournit des caractéristiques qui sont issues de la représentation non supervisée qu'elle se fait du signal. La deuxième est une architecture convolutive avec une représentation supervisée. Nous avons mené une étude empirique afin de jauger la capacité d'une architecture par rapport à une autre à se construire une bonne représentation à partir d'une quantité de données limitée et en mauvais état.

Les résultats obtenus sont les suivants : utilisant des filtres construits avec des algorithmes conçus par des experts, nous avons obtenu sur notre jeu de données un taux de justesse de 89.07%. Puis, en construisant un filtre de manière non supervisée, nous avons obtenu des résultats modestes avec un taux de justesse de 71.8%. Nous avons remédié à ces résultats par l'optimisation d'une liste de paramètres et en changeant la fonction d'initialisation des poids des différentes connexions neuronales dans les architectures étudiées. Nous avons obtenu des résultats plus proche de ceux du filtre choisi par l'expert, avec un taux de justesse de 85%.

Nos conclusions sur ce chapitre sont les suivantes :

— Le compromis entre la quantité de données utilisée lors de l'entraînement et l'utilisation de meilleurs modèles d'apprentissage doit être considéré avec attention.

— Le sous-apprentissage est une problématique aussi importante que le sur apprentissage et celle-ci n'est pas toujours résolue par l'extension du jeu de données.

— Comme perspectives, nous comptons implémenter la fonction ReLU dans les auto-encodeurs et permettre une rétropropagation sur les couches comportant un extremum pooler afin de permettre l'utilisation du *finetuning*.

Chapitre 8

Conclusion et perspectives

Sommaire

8.1 Conclusion **141**
8.2 Perspectives **142**

8.1 Conclusion

Cette thèse était dédiée à la tâche de la classification des textes saisi sur un ordinateur (numérique) ou écrit à la main et ensuite numérisé dans une image. Dans les deux cas, nous avons représenté les textes par une matrice à grande dimension. Nos contributions présentées sous la forme d'algorithmes s'attaquent à deux problématiques : la réduction de la dimension de la matrice en des composantes caractéristiques et l'utilisation des composantes extraites pour améliorer les performances de la tâche de classification.

Nous avons consacré la première partie de la thèse aux textes numérique. Pour réduire la dimension de la matrice de co occurrences nous avons construit un espace sémantique latent, à partir d'un corpus de texte, où nous avons représenté les documents par des vecteurs de probabilités. Nous avons par la suite proposé deux méthodes de classification. Nous avons expliqué cette approche conceptuelle dans le chapitre 3.

Afin de suivre la dynamique des thèmes dans le corpus, nous avons besoin d'un algorithme de classification qui intègre à la fois les thèmes majeurs et les thèmes susceptibles de l'être. La première méthode de classification, présentée dans le chapitre 4, permet donc de regrouper les documents selon deux critères. Le premier est qualitatif et il est représenté par les thèmes majeurs dans la collection des documents. Le deuxième est quantitatif et il est représenté la distance entre la distribution des thèmes dans chaque document de la collection. Au sein de cette méthode, nous avons procédé à l'analyse de la structure de l'espace sémantique latent à l'aide des axiomes de la prétopologie. Ceci nous a permis de fournir, à la première méthode de classification, le nombre de classes ou groupes ainsi que leurs centroïdes respectifs. La méthode que nous avons développée, dans le chapitre 4, nous évite une initialisation aléatoire de ces paramètres et nous permet d'obtenir une meilleure qualité dans les regroupements de documents. Le suivi dynamique des thèmes dans le corpus nécessite la gestion du passage à l'échelle. Nous avons proposé une implémentation

informatique pour distribuer le calcul de l'allocation latente de Dirichlet sur une grille d'ordinateurs.

Dans le chapitre 5, nous avons adressé la problématique de l'archivage des documents textuels en prenant en compte la structure thématique des documents. Ceci a un effet direct sur la phase de la recherche, les documents que l'algorithme a archivés automatiquement au travers une hiérarchie de catégories, sont accessible par des requêtes comportant les thèmes et non pas de simples mots clés. Pour mener ce travail, nous avons développé une version semi-supervisée de l'algorithme de l'allocation latente de Dirichlet et avons combiné sa sortie à l'algorithme des forêts aléatoires. Le résultat de ce mélange est un algorithme de classification capable d'archiver un document dans une hiérarchie de catégories qui commence par les catégories les plus générales vers les catégories les plus particulières.

Nous avons consacré la deuxième partie de la thèse aux textes numérisés. La tâche était de classifier des manuscrits selon le script (langue écrite) qu'ils continent et ce à partir d'une matrice de pixel qui représente l'image. Pour réduire la dimension de la matrice de pixels, nous avons appliqué trois principaux types de filtres pour faire émerger les composantes les plus caractéristiques. Ces composantes sont comme des profils du tracé du script et elles caractérisent soit la direction du tracé, la largeur ou le contraste. Notre approche consistait à prendre l'image de la page du document dans sa globabilité et à extraire des profils qui mélangent les caractéristiques du tracé. Mais aussi, celles de l'environnement où il se trouve par exemple, la qualité du papier, les images, et différents artefacts.

Afin d'extraire des profils mélangeant les caractéristiques du tracé aux caractéristiques de son environnement c'est-à-dire la page, nous avons utilisé les réseaux de neurones de types auto-encodeurs. Les auto-encodeurs nous fournissent des caractéristiques qui sont issues de la représentation non supervisée qu'ils se font du signal. Dans le chapitre 7, nous avons mené une étude empirique où nous avons établi une comparaison avec les caractéristiques extraites avec l'algorithme HOG (en anglais Histogram Oriented Gradient) qui détermine la direction du tracé. Ces caractéristiques sont déterminées par des algorithmes conçus par des experts. Les caractérstiques extraites par les architectures de réseaux de neuronnes que nous avons développé nous ont donné des résultats comparables aux caractéristiques HOG à hauteur de 89% de taux de justesse.

8.2 Perspectives

Deux directions sont esquissées pour nos travaux futurs.

Utilisation des réseaux de neurones récurrents

Dans la première partie de cette thèse, nous avons utilisé des modèles $n-gram$ pour approcher la problématique de la catégorisation des textes numériques. Comme ce type de modèle nécessite une décomposition du texte en terme unique, son application sur des langues non latines peut devenir problématique. Notamment, à cause

de la différence morphologique qu'il peut y avoir avec d'autres langues (e.g. l'arabe et ses différents dialectes).

Des travaux précédents ont montré le potentiel des modèles à caractères (character language model en anglais) et performances supérieures aux modèles $n-gram$. Ce type de modèle peut être appliqué à n'importe quelle langue ou langage écrit par l'utilisation d'un alphabet et ne nécessite pas l'utilisation de termes uniques. Parmi ce type de modèles, les réseaux de neurones récurent de type LSTM (pour Long Short Term Memory) sont d'un intérêt particulier pour nous. Car mal exploité dans l'état de l'art ils peuvent apporter des solutions intéressantes pour l'identification des dialectes arabes.

Utilisation de la fonction Relu et les réseaux de neurones résiduels

En vue des récents résultats obtenus avec des architectures plus profondes 100 à 1000 couches, nous souhaitons dans des travaux futurs explorer ce terrain. Ceci nécessiterait l'utilisation de la fonction Relu au lieu de la fonction softsign. La fonction Relu permet d'éviter l'explosion du gradient ou qu'il atteigne une valeur nulle. Les réseaux à plus d'une centaine de couches ont été récemment étudiés dans les travaux de [He 2015].

Les sujets de nos futurs travaux sont :
- La détection des passages parallèles dans un texte.
- L'identification des auteurs d'un manuscrit avec les auto-encodeurs et les méthodes variationnelles.
- Extraction des lignes dans les manuscrits avec une modification du *seam carving*.

Annexe A

Les défis de l'apprentissage automatique

A.1 Apprentissage supervisé

La situation où se situe l'apprentissage supervisé est la suivante : face à une quantité de données qui est finie, nous souhaitons avoir la capacité d'induire une hypothèse optimale avec un algorithme d'apprentissage s'exécutant en un temps polynomial. Une hypothèse optimale doit être capable de recréer la catégorisation des données $\mathcal{X}$ d'apprentissage avec une erreur minimale et généraliser son apprentissage au niveau des concepts $\mathcal{Y}$ avec une erreur minimale. Selon la nature des concepts $\mathcal{Y}$ la tâche de catégorisation est appelée classification ou régression. Nous limiterons notre synthèse, dans ce chapitre, aux problèmes de classification à deux classes. Il est à noter que les notions étudiées sont extensibles aux cas de plusieurs classes ou dans le cas de variables continues à la régression.

Définition A.1 (Classification)**.** La classification est une opération qui consiste à associer à un ensemble de données $\mathcal{X}_i, 1 \leq i \leq N$ un ensemble de labels $\mathcal{Y}_i, 1 \leq i \leq N$ où chaque label est relatif à une classe. Par exemple, un classifieur binaire h est définit de la façon suivante $h : \mathcal{X}_i, 1 \leq i \leq N \rightarrow \{-1, +1\}$, la sortie de h est qualitative.

Définition A.2 (Regression)**.** La régression est une opération qui consiste à associer un ensemble de données $\mathcal{X}_i, 0 \leq i \leq N$ à un ensemble de sorties quantitatives $\mathcal{Y}_i, 0 \leq i \leq N$ où $\mathcal{Y} \in \mathbb{R}$. Par exemple, prédire la température ou la valeur d'une action en bourse qui sont souvent des variables décrites par un nombre réel. L'analyse de la régression est souvent faite sur une erreur quadratique.

La problématique avec la configuration que nous venons de décrire est la suivante : nous n'avons pas de connaissance sur la provenance exacte de $\mathcal{X}$ et nous n'avons pas la liste des règles à utiliser pour faire apprendre à la machine et lui donner les moyens de reconnaitre les liens insoupçonnés avec les concepts $\mathcal{Y}$. La solution au problème de l'apprentissage, c'est d'abord d'avoir une configuration probabiliste de l'environnement où l'on souhaite que notre machine apprenne. Ensuite, d'approcher le vrai modèle d'apprentissage par l'optimisation d'un modèle approximativement vrai.

Dans ce qui suit, nous décrirons d'abord en nous basant sur une taxonomie détaillée issue de l'état de l'art, la configuration où l'on va situer la problématique du

modèle d'apprentissage. Nous présenterons ensuite, la transformation de la problématique d'origine en une problématique d'optimisation. Nous discuterons enfin, de la faisabilité de l'apprentissage ce qui introduira le besoin des formalismes que l'on présentera par la suite.

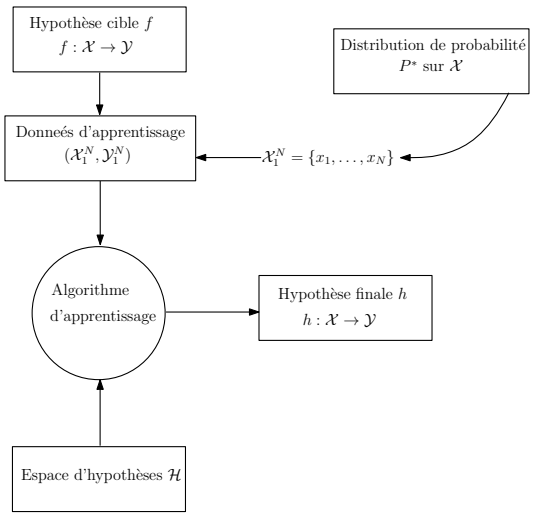


FIGURE A.1 – Diagramme qui décrit le processus de l'apprentissage statistique. Diagramme inspiré des cours du Prof. Yasser Abu Moustafa

Nous nous situons dans le cadre de l'apprentissage supervisé dont le processus est décrit dans le diagramme illustré dans la figure A.1. Ce diagramme est présenté pour introduire un ensemble de taxonomies auxquelles nous nous référerons dans les parties suivantes de ce chapitre.

Nous cherchons à inférer un lien fonctionnel entre les données $\mathcal{X}$ et les concepts $\mathcal{Y}$. Nous nous munissons pour cela d'un échantillon d'apprentissage constitué de la paire $(\mathcal{X}_1^N, \mathcal{Y}_1^N)$. L'échantillon $\mathcal{X}_1^N = \{x_1, \ldots, x_N\}$ est produit par une distribution P qui nous est inconnue. Le lien entre l'échantillon d'exemples $\mathcal{X}_1^N$ et les concepts qui leur sont associés $\mathcal{Y}_1^N$, est donné par l'hypothèse cible, appelée aussi oracle ou superviseur. La fonction cible nous est aussi inconnue.

La recherche du lien fonctionnel se fait par apprentissage où un algorithme est entrainé avec des paires $(\mathcal{X}_1^N, \mathcal{Y}_1^N)$. Le but est de trouver l'algorithme d'apprentissage capable de nous fournir une hypothèse finale h issue de l'espace d'hypothèses $\mathcal{H}$. L'hypothèse h doit être capable de fournir **la meilleure** reproduction des associations $\mathcal{X} \to \mathcal{Y}$ issues de la fonction cible f. La recherche de la meilleure approximation de la fonction cible f dépend de trois facteurs essentiels : la structure de l'espace d'hypothèse $\mathcal{H}$, la structure de l'échantillon d'apprentissage et les critères avec lesquels nous évaluerons l'hypothèse finale h.

configuration probabiliste

Étant donné que, nous n'avons aucune information sur la provenance de $\mathcal{X}$ et $\mathcal{Y}$, et que nous voulons éviter de poser des hypothèses fortes sur la provenance de la fonction cible f pour ne pas affecter l'apprentissage. Nous travaillerons dans un environnent probabiliste en supposons une certaine incertitude à propos des données d'apprentissage constituées de la paire $(\mathcal{X}, \mathcal{Y})$.

$$\mathcal{P}(x, y) = P_{Y|X}(y|X = x)P_X(x) \tag{A.1}$$

Sous cette configuration probabiliste, l'échantillon d'exemples $(\mathcal{X}_1^N, \mathcal{Y}_1^N)$ qui sert à l'entraînement, est tiré avec la distribution $\mathcal{P}(x, y)$, et est considéré comme une variable aléatoire (X, Y). La distribution jointe $\mathcal{P}(x, y)$ est inconnue, où $P_{Y|X}$ est la distribution conditionnelle de Y et P_X est la distribution marginale sur $\mathcal{X}$ (P dans la figure 2.1). L'algorithme d'apprentissage doit fournir l'hypothèse h optimale qui fournira **la meilleure** reproduction des associations $\mathcal{X} \to \mathcal{Y}$. Nous introduirons pour ce fait, un risque relatif au choix de h par rapport à d'autres hypothèses dans $\mathcal{H}$.

Risque

Le but ici est de définir un critère d'évaluation et de l'optimiser afin d'augmenter la qualité d'un modèle d'apprentissage choisi. Le risque R calculera le coût (en bits) pour chaque instance du modèle au moment de son utilisation. Ce coût est défini comme étant numérique et il est plus connu sous le nom de fonction objective. Étant donné la paire de variables aléatoires (X, Y) tirée de la distribution de probabilité P, le risque R s'écrit par rapport à la distribution de probabilité P de la façon suivante :

$$R(h) = \int \frac{1}{2}|Y - f(x, h)|dP(x, y) \tag{A.2}$$

Le problème d'origine est transféré vers un problème d'optimisation de la fonction objective dans le but de minimiser le risque. L'hypothèse optimale s'écrit alors $h^* = \arg\min R(h)$. Cependant, comme P est une distribution de probabilité qui nous est inconnue et R est non explicitement définie, deux solutions s'offrent à nous : la première est de raisonner par induction à partir de l'échantillon des exemples $\mathcal{X}_1^N$ et la deuxième est de restreindre l'espace d'hypothèse $\mathcal{H}$ afin qu'il admette l'hypothèse h la plus proche de l'hypothèse optimale h^*. Identifier h^* pose donc un problème sur la structure de l'espace d'hypothèses $\mathcal{H}$.

A.2 Formalisme de Vapnik et Chervonenkis

La théorie de l'apprentissage statistique développée dans l'ouvrage de Vapnik dans sa courte et longue version [Vapnik 1998] (incluant des démonstrations) donne à l'apprentissage automatique un cadre théorique possible. Dans les travaux de Vapnik et Chervonenkis, le problème de l'apprentissage devient un problème statistique de

sélection de modèle pénalisée (d'où le nom de théorie statistique de l'apprentissage). La sélection de modèle pénalisée n'est pas une approche nouvelle et elle revient aux travaux d'Akaike. L'approche original de Vapnik est d'introduire un concept original de la complexité appelée la dimension Vapnik-Chervonenkis (dimension VC). Nous présentons dans ce qui suit une synthèse de ce dernier.

A.2.1 Cadre général

Les travaux de Vapnik et Chervonenkis [Vapnik 1998] ne sont pas contradictoires avec l'approche PAC et peuvent être considérés comme les travaux fondateurs. La principale différence entre l'apprentissage PAC et la théorie statistique de l'apprentissage est essentiellement une question de vocabulaire. D'un coté avec le formalisme PAC nous avons un vocabulaire orienté vers les informaticiens et de l'autre nous avons un vocabulaire qui est orienté vers les statisticiens.

Sélection de modèles

Nous reprenons dans cette partie, le formalisme de Vapnik pour la sélection de modèles en statistique.

Minimisation du risque empirique (ERM)

Etant donné l'ensemble $\mathcal{H}$ des fonctions de prédictions h, et $\mathcal{X}$ un échantillon d'apprentissage de taille n, $\mathcal{X}_1^N = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$. Nous souhaitons trouver la fonction h qui a le nombre le plus petit d'erreurs. La stratégie la plus commune est d'introduire un risque empirique $R_n(h)$.

Définition A.3. Le risque empirique $R_n(h)$ est construit sur la base de l'échantillon des points $\mathcal{X}_1^N$ où le nombre d'erreurs faites par le classifier h est calculé et par la suite normalisé.

$$R_n(h) = \frac{1}{n} \sum_{i=1}^{n} L(Y_i, h(X_i)) \tag{A.3}$$

Définition A.4. La minimisation du risque empirique, ou ERM pour *Emprical Risk Minimization* est ce que Vapnik appel le principe inductive est le fait de trouver une fonction $h \in \mathcal{H}$ qui minimise $R_n(h)$ ceci se traduit par $h_n = \arg\min_{h \in \mathcal{H}} R_n(h)$. Ainsi, la fonction de prédiction h_n est la meilleure de l'espace d'hypothèse $\mathcal{H}$, et ceci selon les exemples $\mathcal{X}_1^N$.

Il est légitime de croire que lorsque le nombre d'exemples tend vers l'infini, il y a une convergence du risque empirique R_n vers le risque réel R et aussi une proximité entre la fonction h et la fonction optimal f. Dans la théorie de l'apprentissage statistique, nous voulons prouver que l'estimateur R_n est convergent ou consistant (traduit de *consistent estimator* en anglais)

Estimateur convergent

Un estimateur convergent est un estimateur qui tend vers la quantité de probabilité estimée quand la taille de l'échantillon d'exemples $\mathcal{X}_1^N$ augmente. La quantité souhaitée ici est l'erreur réel $\underset{f\in\mathcal{H}}{inf} R(f)$. L'écart $R(h) - \underset{f\in\mathcal{H}}{inf} R(f)$ est la quantité qui nous intéresse. La convergence de l'ERM rendra cette écart faible avec l'augmentation du nombre d'exemples $\mathcal{X}_1^N$. Cette écart peut être étudié dans un ensemble bornée où la vitesse de convergence ne dépend que de la structure de l'ensemble d'hypothèses $\mathcal{H}$. Pour cela nous reconsidérons l'écart précédent par rapport à l'erreur de Bayes R^*. Cette formulation vient des travaux de Vapnik et Chervonenkis [Vapnik 1998] et retrouve racine dans les théorèmes historique de Kolmogorov-Simrnov.

$$R(h) - R^* = \left(R(h) - \underset{f\in\mathcal{H}}{inf} R(f)\right) + \left(\underset{f\in\mathcal{H}}{inf} R(f) - R^*\right) \tag{A.4}$$

$$R(h) - R^* = E_{var} + E_{bias} \tag{A.5}$$

La taille de $\mathcal{H}$ est un compromis. La première quantité dans l'équation (2.3) représente l'erreur d'estimation (variance) et la deuxième quantité représente l'erreur d'approximation (biais). Quand la taille de $\mathcal{H}$ est large nous obtenons une erreur d'estimation E_{var} qui est grande, et une erreur d'approximation E_{bias} qui est petite. À l'inverse si la taille de $\mathcal{H}$ est restreinte nous obtenons une E_{var} qui est petite, et une E_{bias} qui est grande. Le problème que nous avons ici est le suivant, nous pouvons avoir une erreur d'approximation proche de zéro avec un $\mathcal{H}$ large mais on pourrait prendre le mauvaise fonction de prédiction h avec une erreur empirique proche de 0 mais qui n'est applicable que sur l'échantillon d'exemples $\mathcal{X}_1^N$. Ce phénomène s'appelle le sur-apprentissage (*overfitting* en anglais). Nous avons besoin donc d'une fonction de prediction qui évite le sur-apprentissage et qui est capable de généraliser son apprentissage sur un nouveau échantillon d'exemples. Dans le cas idéal nous avons besoin que :

1. $\mathcal{H}$ soit assez "large" pour que l'erreur d'approximation (biais) soit petite.
2. $\mathcal{H}$ soit assez "petit" pour que l'erreur d'estimation (variance) soit petite.

La problématique de la complexité dans l'apprentissage survient suite aux considérations que l'on se fait des conditions du choix de l'espace d'hpothèses $\mathcal{H}$. Cette espace d'hypothèses détermine la convergence de l'estimateur ERM. Nous verrons dans la suite que la dimension de Vapnik-Chervonenkis apparait comme la plus approprié pour résoudre cette problématique.

A.2.2 La dimension de Vapnik et Chervonenkis (VC)

Dans ce qui suit nous définirons la dimension VC dans le cas de la classification binaire. Il ne sera pas question dans cette partie de développer les démonstrations mathématiques de la dimension VC mais de l'introduire comme outils essentiel dans la compréhension de la catégorisation de concepts. Afin de définir cette quantité,

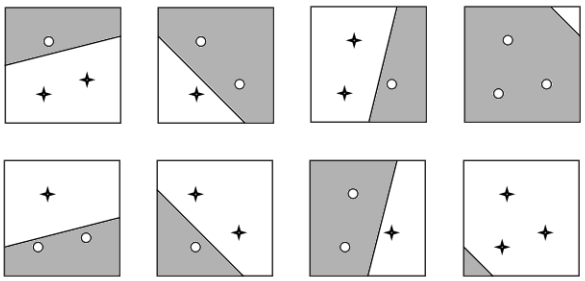


FIGURE A.2 – $m_H(N) = 8$ avec $N = 3$

nous introduisons les définitions de deux concepts fondamentaux : la fonction de croissance (*Growth function* en anglais). Et les espace explosé (*Shattered sets* en anglais).

Fonction de croissance

Une fonction de croissance est une fonction appliquée sur un ensemble de dichotomies. Nous travaillons dans le cas de la classification binaire, une dichotomie est définie donc comme $h : \mathcal{X}_1^N \to \{-1, +1\}$ où $\mathcal{X}_1^N$ est un sous ensemble de $\mathcal{X}$ est sa valeur est $\{x_1, x_2, ..., x_N\}$. Puisque une dichotomie est une hypothèse appliquée à un sous ensemble de $\mathcal{X}$, nous pouvons dénoter le nombre de dichotomies en fonction de l'espace d'hypothèses $\mathcal{H}$ et ceci de la façon suivante : $|\mathcal{H}(x_1, x_2, ..., x_N)|$. Dans ce cas, la question est la suivante : combien pouvons-nous obtenir de classification possible des exemples dans $\mathcal{X}_1^N$ en appliquant différentes dichotomies h ? Le maximum que l'on obtenir dans ce cas est 2^N. Le cardinal maximal des dichotomies possibles s'écrit donc :

$$|\mathcal{H}(x_1, x_2, ..., x_N)| = 2^N \tag{A.6}$$

Définition A.5 (Fonction de croissance)**.** La fonction de croissance compte le nombre maximal de dichotomies que l'on peut appliquer sur un nombre d'exemples N en utilisant l'espace d'hypothèse $\mathcal{H}(x_1, x_2, ..., x_N)$. Elle est dénoté de la façon suivante :

$$m_{\mathcal{H}}(N) = \max_{\mathcal{X}_1^N \in \mathcal{X}} |\mathcal{H}(x_1, x_2, ..., x_N)| \tag{A.7}$$

La fonction de croissance est limité par la quantité 2^N comme suit :

$$m_{\mathcal{H}}(N) \leq 2^N \tag{A.8}$$

Comme exemple de l'application de la définition 1, nous avons illustrer dans la figure A.2 le nombre maximal de dichotomies appliquées sur 3 points. Chaque partie grisée est une dichotomie qui sépare les trois point soit en rond qui correspond (i.e. +1) soit en croix qui correspond à la valeur (i.e. -1).

Dans l'exemple illustrée dans la figure A.2 $m_{\mathcal{H}}(N=3) = 2^3 = 8$. Dans certain cas ceci est possible dans d'autres cas nous avons un nombre de dichotomies inférieur à 2^N. Ce nombre de dichotomies maximal et qui est inférieur à 2^N est appelé point de cassure (*break point* en anglais). Cette notion de point de cassure est importante car elle permet de caractériser un hypothèse ou une dichotomie h.

Définition A.6 (Point de cassure)**.** Un point de cassure k est la taille de $\mathcal{X}_1^k$ pour laquelle il n'existe pas d'espace d'hypothèse $\mathcal{H}(\mathcal{X}_1^k)$ qui peut donner une dichotomie maximal de 2^N. La fonction de croissance s'écrit alors :

$$m_{\mathcal{H}}(k) < 2^k \tag{A.9}$$

Les principaux résultat que l'on obtient avec la définition d'un point de cassure sont présentés dans le lemme 1.

Lemme A.1.
1. Si il n'existe pas de point de cassure k alors $m_{\mathcal{H}}(N) = 2^N$.
2. Si il existe un point de cassure k alors $m_{\mathcal{H}}(N)$ est polynomial en N.

Le deuxième point du lemme 1 est important car si la fonction de croissance est polynomial alors nous pouvons apprendre sur l'espace d'hypothèse $\mathcal{H}$. Et de ce fait, la fonction de croissance peut être réécrite de la manière suivante :

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{k-1} \binom{N}{i} \tag{A.10}$$

Le point de cassure peut être trouver de plusieurs façons soit en prouvant que l'on peut plus avoir de dichotomies à 2^N, soit en prouvant que la fonction de croissance n'est pas une exponentielle et qu'il y aura une cassure. Le point de cassure peut être trouver aussi avec la dimension VC que l'on présentera dans ce qui suit.

Définition A.7 (Dimension VC)**.** La dimension VC est une quantité définie pour une espace d'hypothèse $\mathcal{H}$, elle est dénoté $d_{VC}(\mathcal{H})$. Elle représente la plus large quantité N d'exemples pour laquelle $m_{\mathcal{H}} = 2^N$. En d'autres termes, la dimension VC présente la capacité de l'espace d'hypothèse $\mathcal{H}$ a catégoriser l'ensemble des points N.

Ceci implique, si $N \leq d_{VC}(\mathcal{H})$ alors il existe 2^N dichtomies possible pour séparer les N exemples. Et implique aussi, si $k > d_{VC}(\mathcal{H})$ alors k est un point de cassure pour $\mathcal{H}$.

Définissons la dimension VC en terme de la fonction de croissance $m_{\mathcal{H}}(N)$. Nous savons que la dimension VC est en dessous du point de cassure et en tenons compte de la fonction A.9, la fonction de croissance s'écrit alors

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{d_{VC}} \binom{N}{i} \tag{A.11}$$

"La dimension VC fournit une mesure approximativement similaire, mais plus générale que la mesure $ln|H$ obtenu à partir d'une analyse PAC. La dimension VC peut être appliqué sur des classes de fonctions continues, ce qu'une analyse PAC standard ne peut pas faire. l'apprentissage PAC et la théorie de l'apprentissage statistique ont été connecté par les quatre allemands (Blumer, Ehrengeucht, Haussler et Warmuth en 1989." [Russell 2009]

A.2.3 Séparateurs à Vaste Marge (SVM)

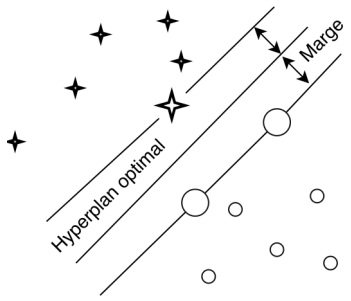


FIGURE A.3 – Un exemple d'un hyperplan optimal qui sépare les données avec une marge maximale

Les séparateurs à vaste marge (SVM) ou encore machines à point de support sont une méthode de classification supervisée binaire. Les SVM ont été introduits par Vapnik dans son ouvrage sur la théorie de l'apprentissage statistique. Conçu comme un classifieur linéaire, la méthode consiste à chercher un hyperplan qui sépare deux classes de données de manière à maximiser la marge entres les exemples les plus proches de chacune des deux différentes classes, ces exemples sont appelés vecteurs support (voir figure 2.3).

Mathématiquement ça consiste à trouver l'hyperplan w avec la marge maximale entre les deux plans qui représenteront les deux classes dans le cas d'une classification binaire. Le problème de la recherche de la marge est écrit avec une formulation de Lagrange et tourné vers un probléme d'optimisation qui est résolu avec l'optimisation quadratique.

La méthode a par la suite été généralisée dans le cas des classes non séparables linéairement avec l'astuce du noyau (Kernel Trick en anglais). L'astuce consiste à faire une projection des exemples dans un espace à grande dimension (infinie) où il est rendu possible de chercher, à travers l'optimisation d'une fonction quadratique un hyperplan optimal pour séparer les exemples.

A.3 Formalisme PAC

Le formalisme PAC (probablement approximativement correcte) a été présenté par Valiant dans son article de 1948 [Valiant 1984]. Valiant fournit un cadre mathé-

matique rigoureux pour analyser la faisabilité de l'apprentissage.

A.3.1 Cadre général

Le formalisme PAC est basé sur le principe de l'induction de l'hypothèse dite vraie (i.e. fonction cible f). L'hypothèse vraie est inférée ou approchée avec la validation de l'espace d'hypothèses $\mathcal{H}$ appelé probablement approximativement correcte. Quant à l'hypothèse dite certainement fausse, elle est découverte avec une forte probabilité, car elle produira une fausse prédiction après un certain nombre d'exemples (i.e. dans le cas de la classification une fausse prédiction se traduit par une fausse association d'un exemples $\mathcal{X}_i$ avec un label $\mathcal{Y}_i$).

Une hypothèse h est dite probablement approximativement correcte est une hypothèse avec une faible probabilité d'être certainement fausse. L'hypothèse probablement approximativement correcte doit être valide sur l'échantillon pour l'apprentissage comme sur celui pour les tests. Pour ce faire, les deux ensembles sont tirés à partir d'une même distribution de probabilité P. La distribution de probabilité P est dite stationnaire, elle ne change pas dans le temps. Ceci assure d'après Valiant la généralisation de l'apprentissage à partir des objets déjà vus, et permet de ce fait de reconnaitre des objets nouveaux.

L'apprentissage PAC est donc une classe restrictive des algorithmes d'apprentissages. Un apprentissage est considéré comme PAC si l'on est capable de reconnaître les irrégularités en un temps polynomial, de réduire le nombre d'interactions avec l'environnement et de réduire l'erreur tant que l'on veut [Valiant 2013]. Avant de formaliser ces dernières assertions, accordons nous d'abord, de nouveau, sur les notations que l'on a vu dans le diagramme de l'apprentissage (Figure 2.1). L'échantillon des paires d'entrainement $(\mathcal{X}, \mathcal{Y})$ est appelé S, il est tiré avec la distribution stationnaire P. S peut être discret ($\in \mathbb{N}$) ou continu ($\in \mathbb{R}$). Dans le cas de la classification, l'hypothèse cible s'écrit $f : S \to \{-1, 1\}$. $\mathcal{H}$ est l'espace des hypothèses. Pour chaque hypothèse $f \in \mathcal{H}$ et chaque exemple $x \in S$,

$$f(x) = \begin{cases} +1 & \text{si } x \text{ est un exemple positif,} \\ -1 & \text{si } x \text{ est un exemple négatif.} \end{cases}$$

Étant donné l'espace d'hypothèses $\mathcal{H}$, et supposé que l'hypothèse cible fait partie de $\mathcal{H}$. Nous définissons l'erreur relative à une hypothèse h comme la probabilité que le résultat de h soit différent du résultat fournit par l'hypothèse cible f étant donné la distribution stationnaire P de l'ensemble des exemples $\mathcal{X}$.

$$erreur(h) = P(h(x) \neq f(x) | x \text{ tirée à partir de } D) \tag{A.12}$$

Nous définirons dans ce qui suit le formalisme PAC autour de la de la faisabilité à savoir : la calculabilité de l'apprentissage en temps polynomial et le nombre polynomial d'exemples pour l'entrainement définissent la faisabilité de l'algorithme d'apprentissage. Le rasoir d'Ockham nous dit de choisir, étant donné S, l'hypothèse

la plus simple, courte et consistante avec $\mathcal{X}$. L'hypothèse la plus courte est l'hypothèse qui aura le plus de capacité à généraliser ses prédictions en vue de nouveaux exemples.

A.3.2 Algorithme PAC-apprenable

Définition A.8. L'espace des hypothèses $\mathcal{H}$ est PAC-apprenable SSi il existe un algorithme d'apprentissage tel que pour chaque $f \in \mathcal{H}$ et ε $(0 < \varepsilon < 1)$, l'algorithme d'apprentissage s'arrête pour un nombre fini d'états et d'exemples. Au moment de l'arrêt l'algorithme d'apprentissage a comme sortie, avec une probabilité $\neq 1 - \varepsilon$, une hypothèse $h \in \mathcal{H}$ qui satisfait la condition suivante :

$$erreur(h) < \varepsilon$$

L'espace des hypothèses $\mathcal{H}$ est polynomialement PAC-apprenable ssi il est PAC-apprenable et que l'algorithme d'apprentissage s'arrête toujours dans un temps et un nombres d'exemples $p(l(f), \frac{1}{\varepsilon})$, pour un polynôme p donné.

Définition A.9. Etant donnée m exemples, et deux constantes α et β tel que $\alpha < 1$ et $\beta \geq 1$. On appelle algorithme d'Ockham, l'algorithme qui s'exécute en un temps polynomial et qui donne en sortie une hypothèse $h \in \mathcal{H}$ en rapport avec les exemples $\mathcal{X}$ et qui satisfait la condition $\mathcal{H}(h) \leq s^{\beta} m^{\alpha}$, avec s égale à la longueur en bits de la plus petite hypothèse appartenant à $\mathcal{H}$.

Théorème A.1. *Un espace d'hypothèse $\mathcal{H}$ est polynomialement PAC-apprenable ssi il existe un algorithme d'Occam. Pour une erreur $0 < \varepsilon < 1$. Choisir m tel que :*

$$m \geq \max\{\left(\frac{2s^{\beta}}{\varepsilon}\right)^{\frac{1}{(1-\alpha)}}, \frac{2}{\varepsilon}\ log\ \frac{1}{\varepsilon}\} \tag{A.13}$$

$\mathcal{H}$ est polynomial en s et $\frac{1}{\varepsilon}$.

Etant donné un nombre m d'exemples satisfaisant la condition dans l'équation 2.4. Dénotons par r le nombre d'hypothèses dans $\mathcal{H}$ et supposons que la fonction cible $f \in \mathcal{H}$. La probabilité que l'hypothèse finale h satisfait la condition $erreur(h) \geq \varepsilon$ (h est une hypothèse fausse) est plus petite que $(1 - \varepsilon)^m r$.

Définition A.10. La taille du plus petit échantillon d'exemples qui satisfait la condition dans l'équation 2.4, est appelé la complexité de l'échantillon de l'algorithme d'apprentissage qui agit sur $\mathcal{H}$.

Selon la définition 2.2 si l'on pose $\alpha = 0$, alors l'on obtient une longueur de l'échantillon de données égale à s^{β}. Selon la définition 2.4, ceci est équivaut à la complexité de l'échantillon d'exemples m. $\mathcal{H}$ est polynomial en s (théorème 2.3), alors l'algorithme d'apprentissage issu de l'échantillon d'exemples s^{β} avec est polynomial et PAC-apprenable.

A.3.3 Le compromis Biais-Variance

Etudions les considérations suivantes, de la structure de l'espace des hypothèses $\mathcal{H}$: nous avons un espace d'hypothèse qui est trop restreint ou pauvre, il est alors possible dans ce cas que l'hypothèse optimale, dite aussi hypothèse vraie, est absente. Même si nous disposons d'une quantité d'exemples infinie pour induire l'hypothèse vraie, nous finirons avec un apprentissage biaisé. Car, nous chercherons à approcher une hypothèse autre que l'hypothèse vraie.

Considérons maintenant, un espace d'hypothèses infini où l'on dispose de toutes les hypothèses possibles, nous aurions certes plus de chance d'approcher l'hypothèse vraie. Cependant, nous avons en notre possession un nombre fini d'exemples pour l'apprentissage. Nous avons dans ce cas, plus de chance d'avoir plusieurs hypothèses qui correspondent à notre jeu d'exemples et donc moins de chance de sélectionner la bonne hypothèse. Des petits changement dans le jeu de données peuvent changer les propriétés fondamentales du modèle d'apprentissage résultant en une grande variance dans la procédure d'apprentissage (appliquer le modèle sur différents jeux d'exemples qui sont générés par l'hypothèse vraie peut donner différents modèles). On dit de ce modèle qu'il a une forte variance ou qu'il est en sur apprentissage.

Ce compromis entre la variance est le biais conditionnera le choix de la taille de notre espace d'hypothèses qui doit permettre de mettre en ouvre un ensemble d'algorithmes d'apprentissage d'une complexité raisonnable.

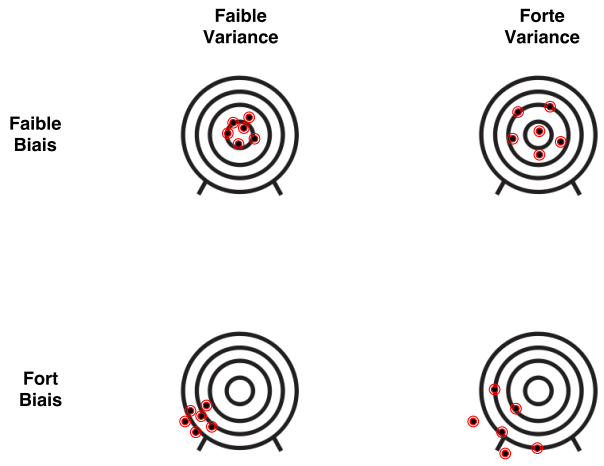


FIGURE A.4 – Schématisation du compromis entre la variance et le bias.

Dans beaucoup de cas, nous préférons un modèle simple que l'on optimise, ou affine, à un modèle complexe qui peut nous induire en erreur avec l'arrivée de nouveaux exemples. Nous appelons aussi le biais, erreur d'approximation. Erreur liée à l'utilisation d'un espace d'hypothèse plutôt que l'espace réel des hypothèses. Nous appelons aussi variance, erreur d'estimation. Erreur liée au fait de trouver une hypothèse qui n'est pas optimale. L'erreur d'approximation est difficilement mesurables vu que l'on connait pas l'espace réel. Par contre, avec l'espace d'hypothèse nous

pouvons mesurer l'erreur d'estimation appelée aussi risque empirique.

Annexe B

Réseaux bayésiens pour la modélisation des textes

B.1 Les réseaux bayésiens

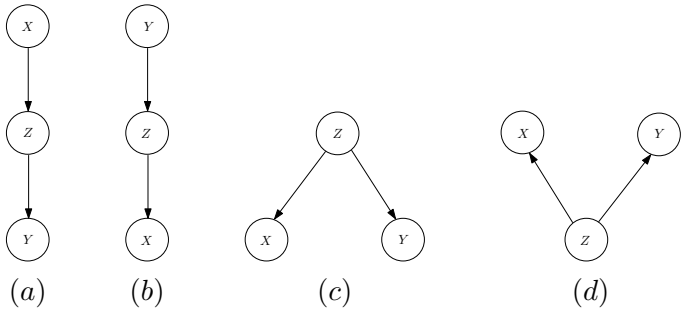


Figure B.1 – Les quatre cas possibles du chemin à deux arcs relient X à Y passant par Z : (a) Effet causal indirect ; (b) Effet probable indirect ; (c) Cause commune ; (d) Effet commun

L'espace de travail des modèles graphiques probabiliste offre essentiellement deux interfaces, qui sont : les réseaux bayésiens et les modèles de Markov cachés. Le réseau bayésien (définition B.1) fournira une structure de données qui permettra de représenter de façon compacte et factorisée une distribution jointe complexe avec des observations *a priori* indépendantes conditionnellement.

Définition B.1. Un réseau bayésien est une structure en graphe orienté acyclique G qui modélise les relations directes ou indirectes entre plusieurs variables aléatoires $\mathcal{X} = X_1, ..., X_n$ représentées par des noeuds.

Définition B.2. Un réseau bayésien G encode les indépendances conditionnelles, appelées aussi indépendance locale en déclarant que chaque noeud X_i est conditionnellement indépendant de ces non-descendants étant donné ses parents (Pa).

$$\forall\ X_i :\ (X_i) \perp (NonDescendants_{X_i} | Pa^G_{X_i}) \tag{B.1}$$

Les dépendances et les indépendances représentées sont des propriétés majeures des distributions jointes et sont indispensables lors du traitement des requêtes : ils réduisent d'une manière considérable le coût du calcul de l'inférence et aide la compréhension du comportement de la distribution. Pour ces raisons, il est important

que notre représentation rende ces propriétés visibles afin que l'expert humain et les algorithmes puissent manipuler la structure des données des réseaux bayésiens. Quatre structures dans le réseau bayésien sont à étudier, ils sont illustrés dans la figure B.1. Dans les quatre cas X, Y sont connectés à travers Z. Les deux figures (a) et (b) représentent une chaine causale (dans les deux directions, la troisième figure (c) une cause commune et la quatrième figure (d) un effet commun. Quand l'influence de X sur la variable Y passe par Z, l'on dit que le chemin $X \rightleftharpoons Z \rightleftharpoons Y$ est actif. L'influence de X sur Y dans les chemins dans les quatre figures de l'illustration B.1 est active dans les cas suivants : (i) le chemin *causal* $X \rightarrow Z \rightarrow Y$, illustré dans la figure (a), B.1 est actif ssi la variable Z n'est pas observée. (ii) Le chemin *probable* $X \leftarrow Z \leftarrow Y$, illustré dans la figure (b), B.1 est actif ssi la variable Z n'est pas observée. (iii) Le chemin $X \leftarrow Z \rightarrow Y$, dit de *cause commune*, illustré dans la figure (c), est actif ssi la variable Z n'est pas observée. (iv) Le chemin $W \rightarrow Z \leftarrow$, dit d'*effet commun*, illustré dans la figure (d), est actif ssi la variable Z ou une de ses descendantes sont observées. Cette structure est appelée aussi *structure en V*.

B.1.1 Raisonnement bayésien

Nous appelons raisonnement bayésien toute approche basée sur la règle de Bayes décrite comme suit :

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})}{p(\mathcal{D})} \tag{B.2}$$

Les données que l'on observe dans notre expérience sont annoté par $\mathcal{D}$, le paramètre θ est une variable aléatoire qui est responsable de la séquence des données $\mathcal{D}$ observée. Nous noterons en gras $\boldsymbol{\theta}$ pour désigner le vecteur des paramètres. Pour faciliter l'approche de l'équation 3.1 nous la réécrivons en terme de proportionnalité, $p(\mathcal{D})$ étant la marginal de vraisemblance et pour simplifier les étapes de dérivation de l'*a posteriori* $p(\boldsymbol{\theta}|\mathcal{D})$ nous l'enlevons et la rajoutons comme une constante de normalisation une fois que la dérivation de l'*a posteriori* est terminée.

$$\textbf{Probabilité a posteriori} \propto \textbf{Vraisemblance} \cdot \textbf{Probabilité a priori} \tag{B.3}$$

Dans l'équation 3.2 nous avons un modèle qui décrit notre expérience et qui est codé dans la **Vraisemblance**, par exemple dans l'expérience de lancée d'une pièce notre modèle sera une distribution de Bernoulli. Nous avons aussi une connaissance acquise au paravent qu'elle soit du domaine où se déroulera notre expérience ou juste une intuition, ceci est codé dans **Probabilité a priori**. Au fur et à mesure que nous voyons de nouvelles données dans le déroulement de notre expérience, nous inférons de nouvelles connaissances, ceci est codé dans la **Probabilité a posteriori**. Les nouvelles connaissances inférées au déroulement de l'expérience deviennent des connaissances *a priori* et ainsi de suite jusqu'à la convergence à l'aide des algorithmes d'approximation de l'inférence comme l'algorithme de Gibbs ou encore des méthodes d'approximation variationnelles (voir sous-section B.1.1). Avec ce type de

raisonnement nous avons donc deux tâches essentielles : la première est celle de l'estimation de paramètre responsable de l'ensemble des données que nous observons et la deuxième est celle de l'inférence de l'a posteriori.

La procédure d'apprentissage bayésien se déroule sur trois étapes, ces étapes présentées d'une manière très générale dans ce qui suit sont applicables à tout type d'apprentissage. Selon sur quoi nous travaillons, notre modèle probabiliste sera différent et les algorithmes qui y sont appliqués seront différents aussi. La procédure de l'apprentissage bayésien permet de mettre à jour des croyances/probabilités sur un ensemble de paramètres qui est responsable des données $\mathcal{D}$ que nous observons.

La procédure de l'apprentissage bayésien

Etape 1

Soit n la séquence d'observation, $x_{1:n} = \{x_1, x_2, ..., x_n\}$, écrire dans un premier temps l'expression relative à la vraisemblance

$$p(x_{1:n}|\theta)$$

Dans cette étape nous spécifions un modèle, les modèles différent d'une application à une autre. Par exemple dans le cas d'une simulation de lancée de pièces notre modèle sera une distribution de Bernoulli, il est spécifier de la manière suivante

$$p(x_{1:n}|\theta) = \theta^m(1-\theta)^{n-m}$$

Etape 2

Spécifier un *a priori* :

$$p(\theta)$$

Dans cette étape nous codons avec une distribution *a priori* nos croyances et l'ensemble des connaissances que nous avons sur le domaine de l'expérience. Dans notre exemple de lancée de pièce notre distribution *a priori* sera une distribution Beta. Nous spécifions la distribution de Beta dans la sous section 4.2.

Etape 3

Calculer l'*a posteriori* :

$$\begin{aligned} p(\theta|x_{1:n}) &= \frac{p(x_{1:n}|\theta)}{p(x_{1:n})} \\ &= \frac{1}{\text{const}}\ p(x_{1:n}|\theta)p(\theta) \\ p(\theta|x_{1:n}) &\propto p(x_{1:n}|\theta)p(\theta) \end{aligned}$$

Dans cette étape nous calculons la distribution *a posteriori* qui est une multiplication de l'*a priori* et de la vraisemblance avec une normalisation de la marginale de la vraisemblance. Cette multiplication et le coeur du raisonnement bayésien, où

l'on met à jour constamment nos croyances et connaissances en ayant de nouvelle observation. L'*a posteriori* devient un *a priori* et ainsi de suite.

À noter la différence avec l'approche fréquentiste qui considère qu'il y' a une seule et unique valeur vraie du paramètre θ et que les données sont des variables aléatoires. L'approche bayésienne considère le paramètre θ comme une variable aléatoire sur laquelle on met une distribution *a priori* et que les données sont des observations qui vont modifier cette distribution *a priori* en la multiplions au modèle que nous avons spécifié.

Spécification de l'apriori

Dans le cas de l'expérience de lancé de pièce notre modèle codé dans la vraisemblance est simple, nous savons que nous variables sont indépendantes mais avec d'autres modèle comme celui de l'allocation latente de Dirichlet le modèle et les connaissances *a priori* prennent rapidement en complexité, l'utilisation des réseaux bayésiens, l'un des modèles graphiques probabiliste avec les chaine de Markov, devient incontournable.

Nous supposons que $X_i \sim Bern(\theta)$, alors $X_i \in \{0, 1\}$. La vraisemblance de la distribution de Bernoulli s'écrit sous la forme suivante

$$p(\mathcal{D}|\theta) = \theta^{N_1}(1-\theta)^{N_0}$$

où $\mathcal{D}$ est la notation de l'ensemble des données qui est synonyme à la notation $x_{1:N}$ et N_1, N_0 correspondent à la somme des variables indicatrices $\mathbb{I}$. Nous avons $N_1 = \sum_{i=1}^{N} \mathbb{I}(x_i = 1)$ faces et $N_0 = \sum_{i=1}^{N} \mathbb{I}(x_i = 0)$ piles. Notons aussi que $0 \leq \theta \leq 1$. θ est la probabilité que $X = 1$ et $1 - \theta$ est la probabilité que $X = 0$. Dans l'interprétation bayésienne les probabilités sont vues comme des degrés de croyance. On appel cela avoir un a priori subjectif. La distribution de probabilité beta nous permet d'encoder notre degré de croyance *a priori* sur la valeur que peut prendre θ. La densité de probabilité de l'*a priori* beta est schématisé dans la figure 1, elle s'écrit sous la forme suivante :

$$Beta(\theta|\alpha_1, \alpha_2) = \frac{1}{B(\alpha_1, \alpha_2)}\theta^{\alpha_1 - 1}(1-\theta)^{\alpha_2 - 1} \tag{B.4}$$

La formule d'inversion de Thomas Bayes, que nous avons écris dans l'équation (1.2) en terme de proportionnalité, permet de mettre à jour les croyances/probabilités sur notre paramètre θ en présence des observations représentées par nos données $\mathcal{D}$. Une nouvelle information est alors obtenue sur θ. Celle-ci peut être utilisée pour mettre à jour notre croyance en une nouvelle valeur de θ et ainsi de suite.

Ce principe est utilisé dans l'inférence dans les réseaux bayésiens et il est établi en multipliant la vraisemblance de Bernoulli par l'*a priori* beta. Quand l'*a posteriori* a la même forme que l'*a priori* nous disons que l'a priori est conjugué à la vraisemblance. Cette notion est beaucoup utilisé dans la littérature des algorithmes d'apprentissage automatique (*Machine Learning algorithms*). L'application

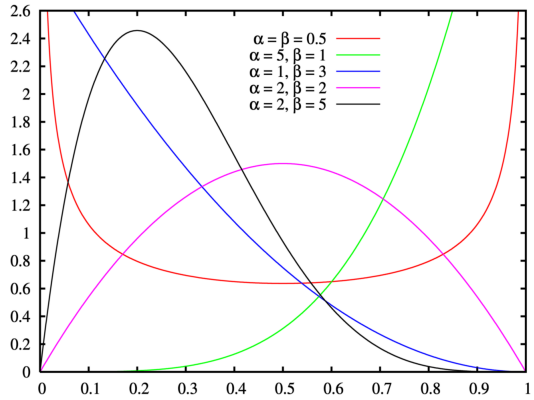


FIGURE B.2 – Densité de probabilité de l'*a prioi* beta

du principe de l'équation (1.2) avec un *a priori* de type beta et une vraisemblance de Bernoulli est écrit dans ce qui suit.

$$\begin{aligned} p(\theta|\mathcal{D}) &\propto p(\mathcal{D}|\theta)p(\theta) \\ &\propto \underbrace{[\theta^{N_1}(1-\theta)^{N_2}]}_{Bernoulli}\underbrace{[\theta^{\alpha_1-1}(1-\theta)^{\alpha_2-1}]}_{Beta} \\ &= \theta^{N_1+\alpha_1-1}(1-\theta)^{N_2+\alpha_2-1} \\ &\propto Beta(\theta|N_1+\alpha_1, N_2+\alpha_2) \end{aligned}$$

Pour une modélisation probabiliste du texte, où chaque document, relatif à un une portion de texte, est un vecteur de variables binaires qui sont à leur tour relatives aux mots dans le document, il nous faut une généralisation de la distribution beta et de la distribution de Bernoulli pour décrire un ensemble de variables binaires et non plus une seule variable binaire. La généralisation de la distribution bêta est ce que l'on appelle une distribution de Dirichlet et la généralisation de la distribution de Bernoulli c'est la multinomial de cette dernière.

Généralisation au cas multinomial

La version multinomial de Bernoulli (*Categorical distribution*) est une instance de la distribution multinomial. Etant donné n instances de x tel que, $x_{1:n} = \{x_1, x_2, ..., x_n\}$. Chaque x_i est relatif à une instance unique composée de K valeurs. Par exemple si $K = 3$ alors les vecteurs possibles sont (100), (010) et (001).

$$x_i = \begin{cases} 1 & 100 \\ 2 & 010 \\ 3 & 001 \end{cases}$$

La vraisemblance des données est

$$p(x_i|\theta) = \prod_{j=1}^{K} \theta_j^{\mathbb{I}^{(x_{ij}=1)}}$$

$$p(x_{1:n}|\theta) = \prod_{i=1}^{n} \prod_{j=1}^{K} \theta_j^{\mathbb{I}^{(x_{ij}=1)}} \tag{B.5}$$

Par exemple pour $i = 3$ nous avons $x_{i=3} = 001$ où $x_{i=3,j=1} = 0, x_{i=3,j=2} = 0$ et $x_{i=3,j=3} = 1$ alors $P(x_{i=3}|\theta) = \theta_1^0 \cdot \theta_2^0 \cdot \theta_3^1 = \theta_3$.

Notons que $P(x_{i=1}|\theta) = \theta_1, P(x_{i=2}|\theta) = \theta_2, P(x_{i=3}|\theta) = \theta_3$ et que $\theta_1+\theta_2+\theta_3 = 1$

Méthodes variationnelles d'approximation

Les méthodes variationnelles d'approximation sont utilisées pour simplifier toute sorte de calculs. L'idée de base consiste à proposer une version réduite du problème original, suffisamment simple pour être utilisé mais ressemblant le plus possible au problème original. Le problème réduit est décrit à l'aide de paramètres variationnels λ qu'on ajuste pour minimiser une fonction de distance D entre le problème d'origine et le problème réduit, souvent en résolvant le système d'équations $\frac{\partial D}{\partial \lambda} = 0$. Dans de nombreux cas, il est possible d'obtenir des bornes inférieures et supérieurs strictes. Les méthodes variationnelles sont utilisées depuis longtemps en statistiques [Rustagi 1976]. En physique statistique, la méthode du champ moyen est une approximation variationnelle particulière dans laquelle les variables individuelles qui constituent le modèle sont supposées complètement indépendantes. Saul et *al.* [Saul 1996] ont développé des fondements mathématiques de l'application de méthodes variationnelles aux réseaux bayésiens et obtenu des approximations exactes des bornes inférieurs pour les réseaux sigmoïdes en utilisant la méthode du champ moyen. Jaakkola et Jordan [Jaakkola 1999] ont étendu la méthodologie pour obtenir aussi bien des bornes inférieurs que supérieurs. Depuis la publication de ces premiers articles, les méthodes variationnelles furent appliquées à de nombreuse familles de modèles, à titre d'exemple nous citons l'excellente thèse [Beal 2003] de Matthew J. Beal sur les algorithmes variationnelles pour l'approximation de l'inférence bayésienne. Dans le cadre de notre travail, ces méthodes facilitent l'approximation de l'inférence des paramètres des modèles probabilistes complexes comme c'est le cas avec l'allocation latente de Dirichlet. Ces méthodes offrent une alternative déterministe et plus rapide comparé aux méthodes de Monte Carlo, dans cette note nous visions à présenter plus précisément une alternative à l'échantillonnage de Gibbs, la méthode de chaine de Markov Monte Carlo la plus utilisée pour l'approximation des paramètres de l'allocation latente de Dirichlet vu sa simplicité d'exécution. Notons que nous les méthodes variationelles d'approximation sont utiles pour l'approche fréquentiste de l'inférence basée sur la maximisation de la vraisemblance comme pour l'approche bayésienne pour l'inférence. Dans cette note nous allons nous intéresser seulement à l'application de ces méthodes pour l'approche bayésienne.

La méthode de Bayes variationnelle

La méthode de Bayes variationelle fait partie des méthodes d'inférence approchées qui, comme la simulation stochastique par Chaîne de Monte-Carlo (méthodes MCMC), offre la possibilité d'économiser le temps de calcul par rapport à des

méthodes d'inférence exactes. Les méthodes d'inférence approchées sont utilisées lorsque la dimension du réseau bayésien augmente, ou lorsque les tables de probabilités conditionnelles ne sont pas exactes. Malgré que les méthode MCMC sont les plus utilisé et plus particulièrement l'algorithme de l'échantillonneur de Gibbs, les méthodes variationnelles sont plus rapide et ont l'avantage d'être peu complexe. Nous détaillons dans ce qui suit la méthode de Bayes variationelle dans le cas de l'inférence bayésienne.

Définition B.3 (Divergence de KL)**.** Solomon Kullback et Richard Leibler, deux cryptanalystes américains sont à l'origine de l'information dite de Kullback-Leibler, une quantité fondamentale qui puise ses ressources dans le concept de l'entropie de Boltzmann . La divergence de KL est une mesure asymétrique de l'éloignement de deux distributions. Elle évalue le nombre de nats (ou de bits si le *log* est en base 2) supplémentaires pour encoder des données provenant de $\mathbb{Q}$ en utilisant un code optimisé pour encoder des données provenant de $\mathbb{P}$. La divergence de KL est définie comme suit :

$$d_{KL}(\mathbb{Q}||\mathbb{P}) = \sum \mathbb{Q}(\boldsymbol{\theta}) \log \frac{\mathbb{Q}(\boldsymbol{\theta})}{\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{X})} \tag{B.6}$$

Dans le cas de l'inférence variationelle, $\mathbb{Q}(\boldsymbol{\theta})$ est la forme simplifiée de la distribution *aposteriori* vraie $\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{X})$. L'inférence est appliquer sur la distribution $\mathbb{Q}(\boldsymbol{\theta})$ qui minimise $d_{KL}(\mathbb{Q}||\mathbb{P})$. Cette méthode est conceptuelement similaire à l'algorithme de la maximisation de l'espérance.

L'inférence bayésienne est basée sur le calcul de la fonction de densité de l'*a posteriori*. Prenons le modèle bayésien générique. Soit $0 \leq \boldsymbol{X} \leq 1$ et $0 \leq \boldsymbol{\theta} \leq 1$ deux distributions de probabilités représentées par un vecteur aléatoire continu.

$$p(\boldsymbol{\theta}|\boldsymbol{X}) = \frac{p(\boldsymbol{X}, \boldsymbol{\theta})}{p(\boldsymbol{X})}$$

Puisque nous travaillerons dans le cas continu nous utiliserons des intégrales. Soit q une fonction de densité de probabilité définie sur l'espace probabilisé Θ. Les méthodes variationelles sont utilisées pour approcher les intégrales nécessaires pour l'inférence bayésienne, pour cela nous appliquons dans ce qui suit une inégalité en introduisant la distribution approchée q. Nous commençons d'abord par appliquer les étapes de l'approche du maximum de vraisemblance en calculer le logarithme de la vraisemblance, ensuite la mise en équation de la différentielle en θ.

Le logarithme de l'intégration de la vraisemblance avec le paramètre θ est le suivant :

$$\log p(\boldsymbol{X}) = \log p(\boldsymbol{X}) \int q(\boldsymbol{\theta}) = \int q(\boldsymbol{\theta}) \log p(\boldsymbol{X}) d\boldsymbol{\theta} \tag{B.7}$$

$$= \int q(\boldsymbol{\theta}) \log \left\{ \frac{p(\boldsymbol{X}, \boldsymbol{\theta})/q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\boldsymbol{X})/q(\boldsymbol{\theta})} \right\} d\boldsymbol{\theta} \tag{B.8}$$

$$= \int q(\boldsymbol{\theta}) \log \left\{ \frac{p(\boldsymbol{X}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})} \right\} d\boldsymbol{\theta} + \int q(\boldsymbol{\theta}) \log \left\{ \frac{q(\boldsymbol{\theta}}{p(\boldsymbol{\theta}|\boldsymbol{X})} \right\} d\boldsymbol{\theta} \tag{B.9}$$

$$\geq \int q(\boldsymbol{\theta}) \log \left\{ \frac{p(\boldsymbol{X}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})} \right\} d\boldsymbol{\theta} \tag{B.10}$$

L'inégalité dans (4) est due au fait que

$$\int q(\boldsymbol{\theta}) \log \left\{ \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\boldsymbol{X})} \right\} d\boldsymbol{\theta} \geq 0 \tag{B.11}$$

Et ce pour toutes les densités q, si et seulement si $q(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\boldsymbol{X})$ L'intégrale dans (5) est connu sous le nom de la divergence de Kullback-Leiber [Kullback 1951] entre q et $p(\cdot|\boldsymbol{X})$(voir Définition 1 dans le cas discret). À partir de (4), il s'ensuit que

$$p(\boldsymbol{X}) \geq p_(\boldsymbol{X}; q),$$

où $p_(\boldsymbol{X}; q)$ est la borne inférieure dépendante de q et qui est donnée par l'intégration de la vraisemblance en $\boldsymbol{\theta}$.

$$p_(\boldsymbol{X}; q) \equiv exp \int q(\boldsymbol{\theta}) \log \left\{ \frac{p(\boldsymbol{X}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})} \right\} d\boldsymbol{\theta} \tag{B.12}$$

L'intérêt de la transformation de densité de l'approche variationelle est l'approximation de la fonction de densité de l'*a posteriori* $p(\boldsymbol{\theta}|\boldsymbol{X})$ avec une autre fonction de densité $q(\boldsymbol{\theta})$ définie sur Θ et pour laquelle la borne inférieure $p_(\boldsymbol{X}; q)$ est plus facile à calculer que $p(\boldsymbol{X})$. En effet en réduisant la fonction de densité de probabilité q à une classe de densité plus réduite en terme de nombre d'opérations de calcul, et puis en maximisant la borne inférieure $p_(\boldsymbol{X}; q)$ sur cette classe de densités nous réduisant la complexité de calcul. Selon (4), maximiser $p_(\boldsymbol{X}; q)$ revient à minimiser la divergence de Kullback-Leibler entre q et $p(\cdot|\boldsymbol{X})$.

Pour restreindre la densité q nous la mettons en facteurs du vecteur des paramètres $\{\theta_1, ..., \theta_M\}$. La factorisation nous donne $\prod_{i=1}^{M} q_i(\theta_i)$. Cette restriction est non paramétriques est elle est issue de la physique statistique connue sous le nom de la méthode du champ moyen. En informatique cette restriction sous la forme de produit ou de factorisation est connue sous le nom de méthode de *Bayes variationelle*. Pour l'inférence sur des données spatiales cette méthode a été comparé à l'échantillonneur de Gibbs est elle s'est vérifié plus rapide dans [Ren 2011]. Les auteurs de ce travail [de Freitas 2013] ont présenté une version amélioré des méthodes MCMC est présenté ici grâce à une combinaison avec les méthodes variationelles.

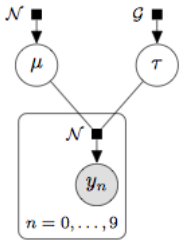


FIGURE B.3 – Modèle graphique probabiliste pour l'exemple de l'inférence des paramètres de la distribution gaussienne avec l'application de la méthode de de Bayes Variationelle. Le modèle est graphe orienté de produits.

B.2 Code de l'implémentation de l'allocation latente de Dirichlet

```
import numpy
import operator
from operator import itemgetter
import codecs

class MCPTM:
    """
    Initiate the first matrices for the computation of the inference
    """
    def __init__(self):
        #In case of french and arabic use the codecs to read or open the file
        f = codecs.open("docs1_notParallel.txt", "r", encoding = "utf-8")
        #f = open("testar.txt")
        self.alpha = 25
        self.beta = 0.01
        self.Vocabdict = dict()
        fline = f.readlines()
        #Liste temporaire pour lire le fichier
        self.tem = []
        #Liste pour identifier tous les mots
        self.words = []
        # a list of list that contains the documents of the corpus
        self.corpus=[]
        # 1. Reading the file formated as following :
        # The first line is the number of
        # documents and the i line represents a documents with a set of words.
        self.vocab = []
        k = 0

        # Number of times word w_i was related with topic z
        self.n_w_i_j = []
        #The number of times all other words are related with topic z
```

```
self.sum_n_i_j = []
#Number of times topic z is related to document d
self.n_z_d = []
#Number of times all other topics are related with document d
self.sum_n_z_d = []

for l in fline:
    self.tem.append(l)
    # Number of topics
self.K = 50
#Number of major topics
self.nbMTP = 10

self.m = 0
# Number of documents
self.m = int(self.tem[0].split()[0])

for i in range(1,self.m+1):
    doc = self.tem[i].split()
    self.corpus.append(doc)



    for j in range(0,len(doc)):
        self.words.append(doc[j])
# Total number of words
self.N = len(self.words)
print self.N
#Extract the unique set of words
self.vocab = set(self.words)
# Number of words in the vocabulary
self.V = len(self.vocab)
print self.V
# topics of words of documents, markov chain

self.z_m_n = []
# word count of each document and topic
self.n_m_z = numpy.zeros((self.m, self.K))
# word count of each topic and vocabulary
self.n_z_t = numpy.zeros((self.K, self.V))
# Sum of each element of the topic count in document
#matrix after each iteration of the Gibbs sampling
self.sum_n_m_z = numpy.zeros((self.m, self.K))
# Sum of each element of the word count in topic matrix after
# each iteraction of the Gibbs sampling
self.sum_n_z_t = numpy.zeros((self.K, self.V))
# word count of each topic
self.n_z = numpy.zeros(self.K)
# Topic distribution in the documents.
```

```
        self.phi = numpy.zeros((self.m, self.K))
        # Word distribution in the topic.
        self.theta = numpy.zeros((self.K, self.V))
        # A tuple that reprsents a document with a the
        # sorted probability for each topic
        self.MTP_matrix = numpy.zeros((self.m,self.nbMTP))


        for i in self.vocab:
            self.Vocabdict [''+i] = k
            k = k + 1

        for i in range(0,self.m):

            z_n = []
            for t in self.corpus[i]:

                z = numpy.random.randint(0, self.K)
                z_n.append(z)
                self.n_m_z[i, z] += 1
                self.n_z_t[z, self.Vocabdict[t]] += 1
                self.n_z[z] += 1

            self.z_m_n.append(numpy.array(z_n))

    #z_m_n is the assignments of the topics for each word
    # in the corpus (in each documents)

    def inference(self):

        for i in range(0,self.m):

            z_n = self.z_m_n[i]

            n_m_z = self.n_m_z[i]

            for n,t in enumerate(self.corpus[i]):


                # discount for n-th word t with topic z
                z = z_n[n]
                #print "the old z %d" %z
                n_m_z[z] -= 1
                self.n_z_t[z, self.Vocabdict[t]] -= 1
                self.n_z[z] -= 1

                # sampling topic new_z for t
                # Number of times word w_i was related with topic z
                self.n_w_i_j = self.n_z_t[:, self.Vocabdict[t]]
```

```
                #The number of times all other words are related with topic z
                self.sum_n_i_j = (self.n_z_t.sum(axis=1) - self.n_w_i_j)
                #Number of times topic z is related to document d
                self.n_z_d = self.n_m_z[i,:]
                #Number of times all other topics are related with document d
                self.sum_n_z_d = (sum(self.n_z_d) - self.n_z_d)

                #Compute the posterior distribution for new assignments

                p_z = ((self.n_w_i_j + self.beta) * (self.n_z_d + self.alpha))
                / ((self.sum_n_i_j + (self.V*self.beta))*(self.sum_n_z_d+(self.K

                #Simulating a multinomial variable for a new topic.

                new_z = numpy.random.multinomial(1,p_z/p_z.sum()).argmax()


                # set z the new topic and increment counters
                z_n[n] = new_z
                n_m_z[new_z] += 1
                self.n_z_t[new_z, self.Vocabdict[t]] += 1
                self.n_z[new_z] += 1
            self.z_m_n[i] = z_n

        print self.n_z_t
        #print self.n_m_z

        self.sum_n_z_t=+self.n_z_t
        self.sum_n_m_z=+self.n_m_z

        # Distribution of the words in each topic. Bayesian estimator.
    def w_t_distribution (self, nbIteration):
        for l in range(0,self.K):
            print "assigning words probabilities to topic ", l
            for w in range(0,self.V):
                self.theta[l][w] = (self.sum_n_z_t[l][w] + self.beta)
                /(sum(self.sum_n_z_t[l,:])+(self.V*self.beta))
        self.w_d_tuples_output(nbIteration)
        return self.theta



    def w_d_tuples_output (self, nbIteration):

        for l in range(0,self.K):

            with open("model_1/theta"+str(nbIteration)+".txt", "a") as myfile:
                myfile.write("topic %d "%l+"\n")
```

```
            w_d_tuple = sorted ([(t,self.theta[l][self.Vocabdict[t]])
                for t in self.vocab], key = itemgetter (1),reverse=True)
            w_d_tuple20 = [w_d_tuple[tup] for tup in range(0,19)]
            with open("model_1/theta"+str(nbIteration)+".txt", "a") as myfile:

                    myfile.write((str(w_d_tuple20)
                        .decode('unicode_escape')).encode('utf-8')+"\n")

#Output of the distribution of the words in the documents
    def w_t_distribution_output(self):
        temp_theta = self.theta

        for l in range(0,self.K):
            with open("theta.txt", "a") as myfile:
                myfile.write("topic %d "%l+"\n")
            for t in self.vocab:
                with open("theta.txt", "a") as myfile:
                    myfile.write(t+" "+
                        str(self.theta[l][self.Vocabdict[t]])+"\n")

#Distribution of the topics in each document. Bayesian estimator.
    def t_d_distribution (self, nbIteration):
        for i in range(0,self.m):
            print "assigning topics probabilities to document ", i
            for l in range(0, self.K):
                self.phi[i][l] = (self.sum_n_m_z[i][l] + self.alpha)
                /(sum(self.sum_n_m_z[i,:])+(self.K*self.alpha))
        self.t_d_tuples_output(nbIteration)
        return self.phi

    def t_d_tuples_output(self, nbIteration):
        temp_phi = self.phi

        for i in range(0, self.m):
            with open("model_1/phi"+str(nbIteration)+".txt", "a") as myfile:
                myfile.write("document %d "%i+"\n")
            t_d_tuple = sorted([(l, self.phi[i][l])
                for l in range(0,self.K)],
                key = itemgetter(1), reverse=True)
            # Extract the major topics of a documents and put it in a matrix
            map_tuple = map(operator.itemgetter(0), t_d_tuple)
            for k in range(0,self.nbMTP):
                self.MTP_matrix[i][k] = map_tuple[k]
            with open("model_1/phi"+str(nbIteration)+".txt", "a") as myfile:
                myfile.write(str(t_d_tuple)+"\n")
        print self.MTP_matrix
        return self.MTP_matrix
```

```
    def multi_relation_data_file (self,nbIteration):
        with open("pretopologyModel/multirelation"+
            str(nbIteration)+".txt", "a") as myfile:
            myfile.write(str(self.K)+"\n")
            myfile.write(' '.join(map(str,[i for i in range(0,self.m)]))+"\n")
            for l in range(0,self.K):
                for i in range(0,self.m):
                    basicN = self.basic_neighberhood(i,l)
                    if basicN == []:
                        with open("pretopologyModel/multirelation"+
                            str(nbIteration)+".txt", "a") as myfile:
                            myfile.write(str(i)+"\n")
                    else:
                        with open("pretopologyModel/multirelation"+
                            str(nbIteration)+".txt", "a") as myfile:
                            myfile.write(' '.join(map(str,basicN))+"\n")


    #Function to compute the basic neighberhood of documents based on the MTP.

    def basic_neighberhood (self,doc,topic):
        basicN = []
        for i in range(0,self.m):
            if i!=doc:
                if topic in self.MTP_matrix[doc][:]
                and topic in self.MTP_matrix[i][:]:
                    basicN.append(i)
        return basicN


    def lda_learning (self, iteration):
        for i in range(iteration):
            print "iteration number ", i
            self.inference()
            self.w_t_distribution(i)
            self.t_d_distribution(i)
            self.multi_relation_data_file(i)


def main ():

    mcptm = MCPTM()
    mcptm.lda_learning(10)


if __name__ == "__main__":
    main()
```

Bibliographie

[Abdul-Mageed 2011] Muhammad Abdul-Mageed, Mona T. Diab et Mohammed Korayem. *Subjectivity and sentiment analysis of modern standard arabic.* In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies : short papers-Volume 2, pages 587–591. Association for Computational Linguistics, 2011. (Cité en page 87.)

[Abdul-Mageed 2014a] Muhammad Abdul-Mageed et Mona Diab. *Sana : A large scale multi-genre, multi-dialect lexicon for arabic subjectivity and sentiment analysis.* Proceedings of the Language Resources and Evaluation Conference (LREC), 2014. (Cité en page 88.)

[Abdul-Mageed 2014b] Muhammad Abdul-Mageed, Mona Diab et Sandra Kübler. *SAMAR : Subjectivity and sentiment analysis for Arabic social media.* Computer Speech & Language, vol. 28, no. 1, pages 20–37, 2014. (Cité en page 88.)

[Aggarwal 2012a] Charu C Aggarwal et ChengXiang Zhai. *A survey of text classification algorithms.* In Mining text data, pages 163–222. Springer, 2012. (Cité en page 2.)

[Aggarwal 2012b] Charu C. Aggarwal et ChengXiang Zhai. *A survey of text classification algorithms.* In Mining text data, pages 163–222. Springer, 2012. (Cité en page 28.)

[Aly 2013] Mohamed A. Aly et Amir F. Atiya. *LABR : A Large Scale Arabic Book Reviews Dataset.* In ACL (2), pages 494–498, 2013. (Cité en page 88.)

[Amdahl 1967] Gene M. Amdahl. *Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities.* In Proceedings of the April 18-20, 1967, Spring Joint Computer Conference, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM. (Cité en page 74.)

[Assael 2016] Yannis M. Assael, Brendan Shillingford, Shimon Whiteson et Nando de Freitas. *LIPNET : END-TO-END SENTENCE-LEVEL LIPREADING.* 2016. (Cité en pages 7 et 12.)

[Bahrampour 2015] Soheil Bahrampour, Naveen Ramakrishnan, Lukas Schott et Mohak Shah. *Comparative Study of Caffe, Neon, Theano, and Torch for Deep Learning.* arXiv preprint arXiv :1511.06435, 2015. (Cité en page 132.)

[Beal 2003] Matthew James Beal. *Variational algorithms for approximate Bayesian inference.* PhD thesis, University of London, 2003. (Cité en pages 36 et 162.)

[Belmandt 2011] ZT Belmandt. Basics of pretopology. Hermann, 2011. (Cité en pages 28, 40, 41 et 42.)

[Bengio 2006] Yoshua Bengio, Pascal Lamblin, Dan Popovici et Hugo Larochelle. *Greedy Layer-Wise Training of Deep Networks.* In NIPS, 2006. (Cité en page 131.)

[Bengio 2007a] Yoshua Bengio, Pascal Lamblin, Dan Popovici et Hugo Larochelle. *Greedy Layer-Wise Training of Deep Networks*. pages 153–160, 2007. (Cité en pages 12 et 111.)

[Bengio 2007b] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle et others. *Greedy layer-wise training of deep networks*. Advances in neural information processing systems, vol. 19, page 153, 2007. (Cité en pages 13, 113, 119 et 133.)

[Bengio 2007c] Yoshua Bengio et Yann LeCun. *Scaling learning algorithms towards AI*. 2007. (Cité en page 110.)

[Blei 2003] David M. Blei, Andrew Y. Ng et Michael I. Jordan. *Latent Dirichlet Allocation*. J. Mach. Learn. Res., vol. 3, pages 993–1022, Mars 2003. (Cité en pages 3, 4, 10, 27, 31 et 33.)

[Blei 2012] David M. Blei. *Probabilistic topic models*. Communications of the ACM, vol. 55, no. 4, pages 77–84, 2012. (Cité en pages ix, 36 et 39.)

[Bluche 2013] T. Bluche, H. Ney et C. Kermorvant. *Feature Extraction with Convolutional Neural Networks for Handwritten Word Recognition*. In 2013 12th International Conference on Document Analysis and Recognition, pages 285–289, Août 2013. (Cité en page 131.)

[Breiman 1996] Leo Breiman. *Bagging predictors*. Machine Learning, vol. 24, no. 2, pages 123–140, Août 1996. (Cité en page 81.)

[Breiman 2001] Leo Breiman. *Random Forests*. Machine Learning, vol. 45, no. 1, pages 5–32, Octobre 2001. (Cité en pages 81 et 118.)

[Cao 2015] Kdd '15 : Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining, New York, NY, USA, 2015. ACM. (Cité en pages 2 et 9.)

[Chen 2015] K. Chen, M. Seuret, M. Liwicki, J. Hennebert et R. Ingold. *Page segmentation of historical document images with convolutional autoencoders*. In 2015 13th International Conference on Document Analysis and Recognition (ICDAR), pages 1011–1015, Août 2015. (Cité en pages 105 et 120.)

[Coates 2010] Adam Coates, Honglak Lee et Andrew Y. Ng. *An analysis of single-layer networks in unsupervised feature learning*. Ann Arbor, vol. 1001, no. 48109, page 2, 2010. (Cité en page 119.)

[Coates 2011] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. J. Wu et A. Y. Ng. *Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning*. In 2011 International Conference on Document Analysis and Recognition, pages 440–445, Septembre 2011. (Cité en page 105.)

[Cooper 1990] Gregory F. Cooper. *The computational complexity of probabilistic inference using bayesian belief networks*. Artificial Intelligence, vol. 42, no. 2-3, pages 393–405, 1990. (Cité en page 36.)

[Dalal 2005] Navneet Dalal et Bill Triggs. *Histograms of oriented gradients for human detection*. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893. IEEE, 2005. (Cité en pages 12, 104 et 115.)

[Dauphin 2014] Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli et Yoshua Bengio. *Identifying and attacking the saddle point problem in high-dimensional non-convex optimization*. arXiv :1406.2572 [cs, math, stat], Juin 2014. arXiv : 1406.2572. (Cité en page 132.)

[de Freitas 2013] Nando de Freitas, Pedro Hojen-Sorensen, Michael I. Jordan et Stuart Russell. *Variational MCMC*. arXiv :1301.2266 [cs, stat], Janvier 2013. arXiv : 1301.2266. (Cité en page 164.)

[Dean 2008] Jeffrey Dean et Sanjay Ghemawat. *MapReduce : Simplified Data Processing on Large Clusters*. Commun. ACM, vol. 51, no. 1, pages 107–113, Janvier 2008. (Cité en page 65.)

[Dorko 2003] G. Dorko et C. Schmid. *Selection of scale-invariant parts for object class recognition*. In Ninth IEEE International Conference on Computer Vision, 2003. Proceedings, pages 634–639 vol.1, Octobre 2003. (Cité en page 105.)

[Du 2015] J. Du, J. F. Zhai, J. S. Hu, B. Zhu, S. Wei et L. R. Dai. *Writer adaptive feature extraction based on convolutional neural networks for online handwritten Chinese character recognition*. In 2015 13th International Conference on Document Analysis and Recognition (ICDAR), pages 841–845, Août 2015. (Cité en page 131.)

[ElSahar 2015] Hady ElSahar et Samhaa R. El-Beltagy. *Building Large Arabic Multi-domain Resources for Sentiment Analysis*. In Alexander Gelbukh, editeur, Computational Linguistics and Intelligent Text Processing, volume 9042, pages 23–34. Springer International Publishing, Cham, 2015. (Cité en page 88.)

[Erhan 2009] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio et Pascal Vincent. *The difficulty of training deep architectures and the effect of unsupervised pre-training*. In International Conference on artificial intelligence and statistics, pages 153–160, 2009. (Cité en pages 131 et 132.)

[Fukushima 1980] Kunihiko Fukushima. *Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*. Biological Cybernetics, vol. 36, no. 4, pages 193–202, Avril 1980. (Cité en page 110.)

[Geman 1984] S. Geman et D. Geman. *Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-6, no. 6, pages 721–741, Novembre 1984. (Cité en page 36.)

[Glorot 2010] Xavier Glorot et Yoshua Bengio. *Understanding the difficulty of training deep feedforward neural networks*. In International conference on ar-

tificial intelligence and statistics, pages 249–256, 2010. (Cité en pages 133 et 134.)

[Gold 1967] E. Mark Gold. *Language identification in the limit.* Information and Control, vol. 10, no. 5, pages 447–474, 1967. (Cité en pages 15 et 16.)

[Griffiths 2004] Thomas L. Griffiths et Mark Steyvers. *Finding scientific topics.* Proceedings of the National academy of Sciences of the United States of America, vol. 101, no. Suppl 1, pages 5228–5235, 2004. (Cité en page 38.)

[Gunther 2008] Neil J. Gunther. *A General Theory of Computational Scalability Based on Rational Functions.* arXiv :0808.1431 [cs], Août 2008. arXiv : 0808.1431. (Cité en page 75.)

[Gunther 2015] Neil J. Gunther, Paul Puglia et Kristofer Tomasette. *Hadoop superlinear scalability.* Communications of the ACM, vol. 58, no. 4, pages 46–55, Mars 2015. (Cité en page 75.)

[Hamamoto 1998] Yoshihiko Hamamoto, Shunji Uchimura, Masanori Watanabe, Tetsuya Yasuda, Yoshihiro Mitani et Shingo Tomita. *A gabor filter-based method for recognizing handwritten numerals.* Pattern Recognition, vol. 31, no. 4, pages 395 – 400, 1998. (Cité en page 104.)

[Harris 1954] Zellig S Harris. *Distributional structure.* Word, vol. 10, no. 2-3, pages 146–162, 1954. (Cité en pages 28 et 31.)

[Hastad 1991] Johan Hastad et Mikael Goldmann. *On the power of small-depth threshold circuits.* computational complexity, vol. 1, no. 2, pages 113–129, Juin 1991. (Cité en page 109.)

[Hastings 1970] W. K. Hastings. *Monte Carlo sampling methods using Markov chains and their applications.* Biometrika, vol. 57, no. 1, pages 97–109, Avril 1970. (Cité en page 36.)

[He 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. *Deep Residual Learning for Image Recognition.* arXiv :1512.03385 [cs], Décembre 2015. arXiv : 1512.03385. (Cité en page 143.)

[He 2017] Sheng He et Lambert Schomaker. *Beyond OCR : Multi-faceted understanding of handwritten document characteristics.* Pattern Recognition, vol. 63, pages 321–333, Mars 2017. (Cité en page 6.)

[Heinrich 2004] Gregor Heinrich. *Parameter estimation for text analysis.* Rapport technique, 2004. (Cité en page 36.)

[Hinton 2006a] Geoffrey E. Hinton et Simon Osindero. *A fast learning algorithm for deep belief nets.* Neural Computation, vol. 18, page 2006, 2006. (Cité en pages 119 et 132.)

[Hinton 2006b] Geoffrey E. Hinton, Simon Osindero et Yee Whye Teh. *A Fast Learning Algorithm for Deep Belief Nets.* Neural Computation, vol. 18, pages 1527–1554, 2006. (Cité en page 131.)

[Ho 1995] Tin Kam Ho. *Random decision forests.* In , Proceedings of the Third International Conference on Document Analysis and Recognition, 1995, volume 1, pages 278–282 vol.1, Août 1995. (Cité en page 80.)

[Hofmann 1999] Thomas Hofmann. *Probabilistic latent semantic analysis.* In Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, pages 289–296. Morgan Kaufmann Publishers Inc., 1999. (Cité en page 31.)

[Hornik 1989] Kurt Hornik, Maxwell Stinchcombe et Halbert White. *Multilayer feedforward networks are universal approximators.* Neural Networks, vol. 2, no. 5, pages 359–366, 1989. (Cité en page 109.)

[Huang 2007] Fu Jie Huang, Y.-Lan Boureau, Yann LeCun et others. *Unsupervised learning of invariant feature hierarchies with applications to object recognition.* In 2007 IEEE conference on computer vision and pattern recognition, pages 1–8. IEEE, 2007. (Cité en page 119.)

[Hubel 1968] D. H. Hubel et T. N. Wiesel. *Receptive fields and functional architecture of monkey striate cortex.* The Journal of Physiology, vol. 195, no. 1, pages 215–243, Mars 1968. (Cité en page 110.)

[Hubert 1985] L. Hubert et P. Arabie. *Comparing partitions.* Journal of classification, vol. 2, no. 1, pages 193–218, 1985. (Cité en page 44.)

[Jaakkola 1999] Tommi Jaakkola et Michael I. Jordan. *Variational Probabilistic Inference and the QMR-DT Network.* JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH, vol. 10, pages 291–322, 1999. (Cité en page 162.)

[Jia 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama et Trevor Darrell. *Caffe : Convolutional Architecture for Fast Feature Embedding.* In Proceedings of the 22Nd ACM International Conference on Multimedia, MM '14, pages 675–678, New York, NY, USA, 2014. ACM. (Cité en pages 132 et 133.)

[Jiang 2010] Chuntao Jiang, Frans Coenen, Robert Sanderson et Michele Zito. *Text classification using graph mining-based feature extraction.* Knowledge-Based Systems, vol. 23, no. 4, pages 302 – 308, 2010. Artificial Intelligence 2009AI-2009The 29th {SGAI} int. conf. on Artificial Intelligence. (Cité en page 31.)

[Kang 2014] Le Kang, Jayant Kumar, Peng Ye, Yi Li et David Doermann. *Convolutional neural networks for document image classification.* In 2014 22nd int. conf. on Pattern Recognition (ICPR), pages 3168–3172. IEEE, 2014. (Cité en page 6.)

[Kearns 1994] Michael J. Kearns et Umesh Vazirani. *An Introduction to Computational Learning Theory*, 1994. (Cité en pages 15, 18 et 19.)

[Klassen 2010] Myungsook Klassen et Nikhila Paturi. *Web document classification by keywords using random forests.* In Networked Digital Technologies, pages 256–261. Springer, 2010. (Cité en pages 78, 83 et 84.)

[Koller 2009] Daphne Koller et Nir Friedman. Probabilistic graphical models : principles and techniques. MIT Press, Cambridge, MA, 2009. (Cité en pages 32, 33, 36 et 80.)

[Krig 2014] Scott Krig. Computer vision metrics : Survey, taxonomy, and analysis. Apress, 2014. (Cité en page 103.)

[Krizhevsky 2012] Alex Krizhevsky, Ilya Sutskever et Geoffrey E. Hinton. *Imagenet classification with deep convolutional neural networks*. In Advances in neural information processing systems, pages 1097–1105, 2012. (Cité en pages 7, 121, 131, 132, 135 et 137.)

[Kullback 1951] S. Kullback et R. A. Leibler. *On Information and Sufficiency*. The Annals of Mathematical Statistics, vol. 22, no. 1, pages 79–86, 03 1951. (Cité en page 164.)

[Landauer 1997] Thomas K. Landauer et Susan T. Dutnais. *A solution to Plato's problem : The latent semantic analysis theory of acquisition, induction, and representation of knowledge*. Psychological review, pages 211–240, 1997. (Cité en pages 3 et 30.)

[Larochelle 2009] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour et Pascal Lamblin. *Exploring strategies for training deep neural networks*. The Journal of Machine Learning Research, vol. 10, pages 1–40, 2009. (Cité en pages 131 et 132.)

[Lecun 1998a] Y. Lecun, L. Bottou, Y. Bengio et P. Haffner. *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, vol. 86, no. 11, pages 2278–2324, Novembre 1998. (Cité en pages 12 et 110.)

[Lecun 1998b] Y. Lecun, L. Bottou, Y. Bengio et P. Haffner. *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, vol. 86, no. 11, pages 2278–2324, Novembre 1998. (Cité en pages 19 et 23.)

[LeCun 2012] Yann LeCun. *Learning invariant feature hierarchies*. In European Conference on Computer Vision, pages 496–505. Springer, 2012. (Cité en pages 119 et 121.)

[Levorato 2008] Vincent Levorato et Marc Bui. *Data structures and algorithms for pretopology : the java based software library pretopolib*. In (I2CS), pages 122–134, 2008. (Cité en page 53.)

[Liparas 2014] Dimitris Liparas, Yaakov HaCohen-Kerner, Anastasia Moumtzidou, Stefanos Vrochidis et Ioannis Kompatsiaris. *News Articles Classification Using Random Forests and Weighted Multimodal Features*. In David Lamas et Paul Buitelaar, editeurs, Multidisciplinary Information Retrieval, volume 8849, pages 63–75. Springer International Publishing, Cham, 2014. (Cité en page 78.)

[Lowe 2004] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints*. Int. J. Comput. Vision, vol. 60, no. 2, pages 91–110, Novembre 2004. (Cité en page 104.)

[Lu 2013] Youwei Lu, Shogo Okada et Katsumi Nitta. *Semi-supervised Latent Dirichlet Allocation for Multi-label Text Classification*. In Moonis Ali, Tibor Bosse, Koen V. Hindriks, Mark Hoogendoorn, Catholijn M. Jonker et Jan Treur, editeurs, Recent Trends in Applied Artificial Intelligence, numéro 7906 de Lecture Notes in Computer Science, pages 351–360. Springer Berlin Heidelberg, 2013. (Cité en page 78.)

[Malmasi 2015] Shervin Malmasi, Eshrag Refaee et Mark Dras. *Arabic Dialect Identification using a Parallel Multidialectal Corpus*. In Proceedings of the 14th Conference of the Pacific Association for Computational Linguistics (PACLING 2015), Bali, Indonesia, 2015. (Cité en page 87.)

[Manning 2008] Christopher D. Manning, Prabhakar Raghavan et Hinrich Schütze. Introduction to Information Retrieval. Cambridge University Press, Juillet 2008. (Cité en page 28.)

[Manning 2009] Christopher D. Manning et Prabhakar Raghavan. *An Introduction to Information Retrieval*, 2009. (Cité en page 2.)

[McCallum 2002] Andrew Kachites McCallum. *MALLET : A Machine Learning for Language Toolkit*, 2002. (Cité en pages x et 73.)

[mer 2015] *mertterzihan/pymc, https ://github.com/mertterzihan/pymc, 2015-07-02*, 2015. (Cité en page 66.)

[Metropolis 1953] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller et Edward Teller. *Equation of State Calculations by Fast Computing Machines*. The Journal of Chemical Physics, vol. 21, no. 6, pages 1087–1092, Juin 1953. (Cité en page 36.)

[Minetto 2013] Rodrigo Minetto, Nicolas Thome, Matthieu Cord, Neucimar J. Leite et Jorge Stolfi. *T-HOG : An effective gradient-based descriptor for single line text regions*. Pattern Recognition, vol. 46, no. 3, pages 1078 – 1090, 2013. (Cité en page 115.)

[Ming 2014] Zijian Ming, Chunjie Luo, Wanling Gao, Rui Han, Qiang Yang, Lei Wang et Jianfeng Zhan. *Bdgs : A scalable big data generator suite in big data benchmarking*. In Advancing Big Data Benchmarks, pages 138–154. Springer, 2014. (Cité en page 65.)

[Minka 2000] Thomas P. Minka. *Estimating a Dirichlet distribution*. Rapport technique, 2000. (Cité en page 35.)

[Mohammad 2015] Saif M. Mohammad, Mohammad Salameh et Svetlana Kiritchenko. *How Translation Alters Sentiment*. Journal of Artificial Intelligence Research, vol. 54, pages 1–20, 2015. (Cité en page 87.)

[Monroe 2014] Will Monroe, Spence Green et Christopher D. Manning. *Word segmentation of informal Arabic with domain adaptation*. ACL, Short Papers, 2014. (Cité en page 87.)

[Mu 2008] Yadong Mu, Shuicheng Yan, Yi Liu, T. Huang et Bingfeng Zhou. *Discriminative local binary patterns for human detection in personal album*. In IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008, pages 1–8, Juin 2008. (Cité en page 104.)

[Mukherjee 2008] Indraneel Mukherjee et David M. Blei. *Relative Performance Guarantees for Approximate Inference in Latent Dirichlet Allocation*. In Proceedings of the 21st International Conference on Neural Information Processing Systems, NIPS'08, pages 1129–1136, USA, 2008. Curran Associates Inc. (Cité en page 64.)

[Nabil 2015] Mahmoud Nabil, Mohamed Aly et Amir F. Atiya. *ASTD : Arabic Sentiment Tweets Dataset.* In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2515–2519, 2015. (Cité en page 88.)

[Nair 2010] Vinod Nair et Geoffrey E. Hinton. *Rectified Linear Units Improve Restricted Boltzmann Machines.* In ICML, 2010. (Cité en page 131.)

[Nallapati 2007] Ramesh Nallapati, William Cohen et John Lafferty. *Parallelized Variational EM for Latent Dirichlet Allocation : An Experimental Evaluation of Speed and Scalability.* In Proceedings of the Seventh IEEE International Conference on Data Mining Workshops, ICDMW '07, pages 349–354, Washington, DC, USA, 2007. IEEE Computer Society. (Cité en page 64.)

[Newman 2009] David Newman, Arthur Asuncion, Padhraic Smyth et Max Welling. *Distributed algorithms for topic models.* The Journal of Machine Learning Research, vol. 10, pages 1801–1828, 2009. (Cité en pages 64, 66 et 67.)

[Pan 2010] S. J. Pan et Q. Yang. *A Survey on Transfer Learning.* IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pages 1345–1359, Octobre 2010. (Cité en page 130.)

[Pasha 2014] Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow et Ryan Roth. *MADAMIRA : A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic.* In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk et Stelios Piperidis, editeurs, Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA). (Cité en page 87.)

[Patil 2012] Ajay S Patil et BV Pawar. *Automated classification of web sites using Naive Bayesian algorithm.* In Proceedings of the International MultiConference of Engineers and Computer Scientists, volume 1, 2012. (Cité en page 78.)

[Poibeau 2016] Thierry Poibeau et Shravan Vasishth. *Introduction : Cognitive Issues in Natural Language Processing*, 2016. (Cité en page 10.)

[Qiu 2014] Zhuolin Qiu, Bin Wu, Bai Wang et Le Yu. *Gibbs Collapsed Sampling for Latent Dirichlet Allocation on Spark.* In Journal of Machine Learning Research, pages 17–28, 2014. (Cité en pages 64 et 66.)

[Ramage 2009] Daniel Ramage, David Hall, Ramesh Nallapati et Christopher D. Manning. *Labeled LDA : A supervised topic model for credit attribution in multi-labeled corpora.* In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing : Volume 1-Volume 1, pages 248–256. Association for Computational Linguistics, 2009. (Cité en page 78.)

[Ranzato 2007] Marc Ranzato, Christopher Poultney, Sumit Chopra et Yann L. Cun. *Efficient Learning of Sparse Representations with an Energy-Based*

Model. In P. B. Schölkopf, J. C. Platt et T. Hoffman, editeurs, Advances in Neural Information Processing Systems 19, pages 1137–1144. MIT Press, 2007. (Cité en pages 12 et 111.)

[Refaee 2014] Eshrag Refaee et Verena Rieser. *An Arabic twitter corpus for subjectivity and sentiment analysis*. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC ?14), Reykjavik, Iceland, may. European Language Resources Association (ELRA), 2014. (Cité en page 88.)

[Ren 2011] Qian Ren, Sudipto Banerjee, Andrew O. Finley et James S. Hodges. *Variational Bayesian methods for spatial data analysis*. Computational Statistics & Data Analysis, vol. 55, no. 12, pages 3197–3217, Décembre 2011. (Cité en page 164.)

[Resnik 2010] Philip Resnik et Eric Hardisty. *Gibbs sampling for the uninitiated*. Rapport technique, DTIC Document, 2010. (Cité en page 37.)

[Rokach 2009] Lior Rokach. *Ensemble-based classifiers*. Artificial Intelligence Review, vol. 33, no. 1-2, pages 1–39, Novembre 2009. (Cité en page 80.)

[Russell 2009] Stuart Russell et Peter Norvig. Artificial Intelligence : A Modern Approach. Prentice Hall, Upper Saddle River, 3 edition édition, Décembre 2009. (Cité en page 152.)

[Rustagi 1976] Rustagi. Variational methods in statistics. Academic Press, Mars 1976. (Cité en page 162.)

[Sahu 2015] D. K. Sahu et C. V. Jawahar. *Unsupervised feature learning for optical character recognition*. In 2015 13th International Conference on Document Analysis and Recognition (ICDAR), pages 1041–1045, Août 2015. (Cité en pages 105 et 120.)

[Saidani 2015] Asma Saidani, Afef Kacem et Abdel Belaid. *Arabic/Latin and Machine-printed/Handwritten Word Discrimination using HOG-based Shape Descriptor*. ELCVIA Electronic Letters on Computer Vision and Image Analysis, vol. 14, no. 2, page 1, Septembre 2015. (Cité en pages xi, 116 et 117.)

[Salameh 2015] Mohammad Salameh, Saif M. Mohammad et Svetlana Kiritchenko. *Sentiment after translation : A case-study on arabic social media posts*. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, pages 767–777, 2015. (Cité en page 87.)

[Salzberg 1994] Steven L. Salzberg. *C4.5 : Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993*. Machine Learning, vol. 16, no. 3, pages 235–240, Septembre 1994. (Cité en page 92.)

[Saul 1996] Lawrence K. Saul, Tommi Jaakkola et Michael I. Jordan. *Mean Field Theory for Sigmoid Belief Networks*. Journal of Artificial Intelligence Research, vol. 4, pages 61–76, 1996. (Cité en page 162.)

[Schmidhuber 2015] Juergen Schmidhuber. *Deep Learning in Neural Networks : An Overview*. Neural Networks, vol. 61, pages 85–117, Janvier 2015. arXiv : 1404.7828. (Cité en page 106.)

[Seuret 2016] Mathias Seuret, Michele Alberti et Marcus Liwicki. *N-light-N*. Rapport technique, Université de Fribourg, 2016. (Cité en page 120.)

[Shaalan 2014] Khaled Shaalan. *A survey of Arabic named entity recognition and classification*. Computational Linguistics, vol. 40, no. 2, pages 469–510, 2014. (Cité en page 87.)

[Shi 2016] Baoguang Shi, Xiang Bai et Cong Yao. *Script identification in the wild via discriminative convolutional neural network*. Pattern Recognition, vol. 52, pages 448–458, Avril 2016. (Cité en page 131.)

[Shin 2016] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura et Ronald M. Summers. *Deep convolutional neural networks for computer-aided detection : CNN architectures, dataset characteristics and transfer learning*. IEEE transactions on medical imaging, vol. 35, no. 5, pages 1285–1298, 2016. (Cité en page 138.)

[Smola 2010] Alexander Smola et Shravan Narayanamurthy. *An architecture for parallel topic models*. Proceedings of the VLDB Endowment, vol. 3, no. 1-2, pages 703–710, 2010. (Cité en page 65.)

[Smyth 2009] Padhraic Smyth, Max Welling et Arthur U. Asuncion. *Asynchronous Distributed Learning of Topic Models*. In D. Koller, D. Schuurmans, Y. Bengio et L. Bottou, editeurs, Advances in Neural Information Processing Systems 21, pages 81–88. Curran Associates, Inc., 2009. (Cité en page 65.)

[Sontag 2011] David Sontag et Dan Roy. *Complexity of Inference in Latent Dirichlet Allocation*. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira et K. Q. Weinberger, editeurs, Advances in Neural Information Processing Systems 24, pages 1008–1016. Curran Associates, Inc., 2011. (Cité en page 67.)

[Srivastava 2014] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever et Ruslan Salakhutdinov. *Dropout : a simple way to prevent neural networks from overfitting*. Journal of Machine Learning Research, vol. 15, pages 1929–1958, 2014. (Cité en page 131.)

[Steinbach 2004] Michael Steinbach, Levent Ertöz et Vipin Kumar. *The challenges of clustering high dimensional data*. In New Directions in Statistical Physics, pages 273–309. Springer, 2004. (Cité en page 29.)

[Steyvers 2007] Mark Steyvers et Tom Griffiths. *Probabilistic topic models*. Handbook of latent semantic analysis, vol. 427, no. 7, pages 424–440, 2007. (Cité en pages 35 et 38.)

[Sutskever 2013a] Ilya Sutskever. *Training recurrent neural networks*. PhD thesis, University of Toronto, 2013. (Cité en pages 18 et 132.)

[Sutskever 2013b] Ilya Sutskever, James Martens, George Dahl et Geoffrey Hinton. *On the importance of initialization and momentum in deep learning*. In Pro-

ceedings of the 30th international conference on machine learning (ICML-13), pages 1139–1147, 2013. (Cité en pages 132 et 135.)

[Tanenbaum 1994] Andrew S. Tanenbaum. Distributed Operating Systems. Pearson, Englewood Cliffs, N.J, 1 edition édition, Septembre 1994. (Cité en page 65.)

[Thakur 2005] Rajeev Thakur, Rolf Rabenseifner et William Gropp. *Optimization of Collective Communication Operations in MPICH*. Int. J. High Perform. Comput. Appl., vol. 19, no. 1, pages 49–66, Février 2005. (Cité en page 65.)

[Thanh Van Le 2007] Nadia Kabachi Thanh Van Le et Michel Lamure. *A clustering method associating pretopological concepts and k-means algorithm*. The 12th International Conference on Applied Stochastic Models and Data Analysis, 2007. (Cité en page 60.)

[Torralba 2011] Antonio Torralba et Alexei A. Efros. *Unbiased look at dataset bias*. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 1521–1528. IEEE, 2011. (Cité en page 105.)

[Torres 2009] Guadalupe J. Torres, Ram B. Basnet, Andrew H. Sung, Srinivas Mukkamala et Bernardete M. Ribeiro. *A similarity measure for clustering and its applications*. Int J Electr Comput Syst Eng, vol. 3, no. 3, pages 164–170, 2009. (Cité en page 61.)

[Valiant 1984] Leslie G. Valiant. *A theory of the learnable*. Communications of the ACM, vol. 27, no. 11, pages 1134–1142, 1984. (Cité en pages 15, 16, 18 et 152.)

[Valiant 2013] Leslie G. Valiant. Probably Approximately Correct : Nature's Algorithms for Learning and Prospering in a Complex World. Basic Books, New York, 2013. (Cité en page 153.)

[Vapnik 1998] Vladimir N. Vapnik. Statistical Learning Theory. Wiley-Blackwell, New York, Octobre 1998. (Cité en pages 15, 16, 118, 147, 148 et 149.)

[Vincent 2008] Pascal Vincent, Hugo Larochelle, Yoshua Bengio et Pierre-Antoine Manzagol. *Extracting and Composing Robust Features with Denoising Autoencoders*. In Proceedings of the 25th International Conference on Machine Learning, ICML '08, pages 1096–1103, New York, NY, USA, 2008. ACM. (Cité en page 130.)

[Vincent 2010] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio et Pierre-Antoine Manzagol. *Stacked denoising autoencoders : Learning useful representations in a deep network with a local denoising criterion*. The Journal of Machine Learning Research, vol. 11, pages 3371–3408, 2010. (Cité en page 133.)

[Vinh 2010] Nguyen Xuan Vinh, Julien Epps et James Bailey. *Information Theoretic Measures for Clusterings Comparison : Variants, Properties, Normalization and Correction for Chance*. J. Mach. Learn. Res., vol. 11, pages 2837–2854, Décembre 2010. (Cité en page 44.)

[Wallach 2009] Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov et David Mimno. *Evaluation methods for topic models.* In Proceedings of the 26th Annual International Conference on Machine Learning, pages 1105–1112. ACM, 2009. (Cité en page 45.)

[Wang 2009] Yi Wang, Hongjie Bai, Matt Stanton, Wen-Yen Chen et Edward Y. Chang. *Plda : Parallel latent dirichlet allocation for large-scale applications.* In Algorithmic Aspects in Information and Management, pages 301–314. Springer, 2009. (Cité en page 65.)

[Wei 2015] Hao Wei, Mathias Seuret, Kai Chen, Andreas Fischer, Marcus Liwicki et Rolf Ingold. *Selecting Autoencoder Features for Layout Analysis of Historical Documents.* In Proceedings of the 3rd International Workshop on Historical Document Imaging and Processing, HIP '15, pages 55–62, New York, NY, USA, 2015. ACM. (Cité en pages 105 et 120.)

[Yan 2002] Xifeng Yan et Jiawei Han. *gspan : Graph-based substructure pattern mining.* In Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE int. conf. on, pages 721–724. IEEE, 2002. (Cité en page 31.)

[Yilmaz 2011] M. B. Yilmaz, B. Yanikoglu, C. Tirkaz et A. Kholmatov. *Offline signature verification using classifier combination of HOG and LBP features.* In Biometrics (IJCB), 2011 International Joint Conference on, pages 1–7, Oct 2011. (Cité en page 115.)

[Zaharia 2010] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker et Ion Stoica. *Spark : cluster computing with working sets.* In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, pages 10–10, 2010. (Cité en page 65.)

[Zhang 2004] Nevin L. Zhang. *Hierarchical latent class models for cluster analysis.* The Journal of Machine Learning Research, vol. 5, pages 697–723, 2004. (Cité en page 32.)

[Zhu 2012] Xiangxin Zhu, Carl Vondrick, Deva Ramanan et Charless Fowlkes. *Do We Need More Training Data or Better Models for Object Detection ?.* In BMVC, volume 3, page 5. Citeseer, 2012. (Cité en page 138.)

Classification du texte numérique et numérisé. Approche basée sur les algorithmes d'apprentissage automatique.

Résumé : Différentes disciplines des sciences humaines telles la philologie ou la paléographie font face à des tâches complexes, fastidieuses et consommatrices en temps pour l'examen des sources de données. La proposition d'approches computationnelles en humanités permet d'adresser les problématiques rencontrées telles que la lecture, l'analyse et l'archivage de façon systématique. Les modèles conceptuels élaborés reposent sur des algorithmes et ces derniers donnent lieu à des implémentations informatiques qui automatisent ces tâches fastidieuses.
La première partie de la thèse vise d'une part, à établir la structuration thématique d'un corpus, en construisant des espaces sémantiques de grande dimension, ou encore à proposer le suivi des thèmes. D'autre part, le suivi dynamique des thématiques constitue un réel défi scientifique, notamment en raison du passage à l'échelle. Une contribution de cette thèse concerne la distribution de ces calculs sur une grille d'ordinateurs, avec des outils tels que Spark, Hadoop et MapReduce.
La seconde partie de la thèse traite de manière holistique la page d'un document numérisé sans aucune intervention préalable. Le but est d'apprendre automatiquement des représentations du trait de l'écriture ou du tracé d'un certain script (langue écrite) par rapport au tracé d'un autre script. Il faut dans ce cadre tenir compte de l'environnement où se trouve le tracé : image, artefact, bruits dus à la détérioration de la qualité du papier, etc. Notre approche propose un empilement de réseaux de neurones auto-encodeurs afin de fournir une représentation alternative des données reçues en entrée. Les résultats obtenus avec les représentations apprises par les réseaux de neurones sont comparables à ceux fabriqués par des experts. De plus, nous proposons un retour d'expériences autour de l'initialisation des paramètres d'apprentissage en vue de pistes d'améliorations futures.
Les contributions de ce travail concernent la conception d'algorithmes pour la classification des textes numériques et numérisés. La classification des textes numériques est opérée à l'aide d'un modèle basé sur des espaces sémantiques quant à la classification des textes numérisée, elle s'appuie sur le développement d'architectures de réseaux de neurones.

Mots clés : Modèle graphique probabiliste, systèmes d'apprentissage, recherche d'information, Apprentissage automatique, apprentissage automatique des caractéristiques, réseaux de neurones.