

Enrichissement des modèles de classification de textes représentés par des concepts

Jean-Charles Risch

▶ **To cite this version:**

Jean-Charles Risch. Enrichissement des modèles de classification de textes représentés par des concepts. Traitement du texte et du document. Université de Reims Champagne-Ardenne, 2017. Français. NNT : . tel-02883213

HAL Id: tel-02883213
https://hal.science/tel-02883213

Submitted on 28 Jun 2020

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 DE REIMS CHAMPAGNE–ARDENNE

ECOLE DOCTORALE SCIENCES TECHNOLOGIE SANTÉ (547)

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

***Discipline :* INFORMATIQUE**

***Spécialité :* INTELLIGENCE ARTIFICIELLE**

présentée et soutenue publiquement par

Jean-Charles RISCH

le 27 Juin 2017

Enrichissement des Modèles de Classification de Textes Représentés par des Concepts

---

Thèse dirigée par **M. Francis Rousseaux**, **Université de Reims Champagne-Ardenne**

Et par **M. Eddie Soulier**, **Université De Technologie de Troyes**

JURY

M. FRANCIS ROUSSEAUX	, Directeur de thèse	, Professeur	, URCA
M. EDDIE SOULIER	, Co-directeur de thèse	, Professeur	, UTT
M. ARNAUD MARTIN	, Président et Rapporteur	, Professeur	, IRISA
M. GILLES KASSEL	, Rapporteur	, Professeur	, UPJV
Mme COLETTE FAUCHER	, Examinateur	, Professeur	, UPMC
M. HACENE FOUCHAL	, Examinateur	, Professeur	, URCA

"We can only see a short distance ahead, but we can see plenty there that needs to be done.."

Alan Turing

Résumé

La majorité des méthodes de classification de textes utilisent le paradigme du sac de mots pour représenter les textes. Pourtant cette technique pose différents problèmes sémantiques : certains mots sont polysémiques, d'autres peuvent être des synonymes et être malgré tout différenciés, d'autres encore sont liés sémantiquement sans que cela soit pris en compte et enfin, certains mots perdent leur sens s'ils sont extraits de leur groupe nominal. Pour pallier ces problèmes, certaines méthodes ne représentent plus les textes par des mots mais par des concepts extraits d'une ontologie de domaine, intégrant ainsi la notion de sens au modèle. Les modèles intégrant la représentation des textes par des concepts restent peu utilisés à cause des résultats peu satisfaisants. Afin d'améliorer les performances de ces modèles, plusieurs méthodes ont été proposées pour enrichir les caractéristiques des textes à l'aide de nouveaux concepts extraits de bases de connaissances. Mes travaux donnent suite à ces approches en proposant une étape d'enrichissement des modèles à l'aide d'une ontologie de domaine associée. J'ai proposé deux mesures permettant d'estimer l'appartenance aux catégories de ces nouveaux concepts. A l'aide de l'algorithme du classifieur naïf Bayésien, j'ai testé et comparé mes contributions sur le corpus de textes labéllisés Ohsumed et l'ontologie de domaine Disease Ontology. Les résultats satisfaisants m'ont amené à analyser plus précisément le rôle des relations sémantiques dans l'enrichissement des modèles. Ces nouveaux travaux ont été le sujet d'une seconde expérience où il est question d'évaluer les apports des relations hiérarchiques d'hyperonymie et d'hyponymie.

Remerciements

J'aimerais remercier en premier lieu mon collègue, Jean Petit, pour avoir été mon binôme durant ces trois années de thèse. Même si nous ne travaillions pas sur le même sujet, nous avons fait face ensemble aux mêmes problèmes qu'un doctorant peut rencontrer...

Je remercie également tout particulièrement mon directeur industriel, Jean Brunet, pour son implication et ses bons conseils que je continue à suivre au delà du cadre de la thèse. Je remercie aussi Catherine Lhermet, l'assistante du skill Capgemini auquel j'appartenais, pour sa sympathie et ses solutions multiples à chaque petits (ou gros) problèmes. De manière plus générale, je remercie l'ensemble des collègues avec qui j'ai pu échanger et apprendre au sein de l'entreprise Capgemini.

Je souhaite remercier mon ami Raphaël Lallement qui a été mon coach de thèse pendant ces trois années. Merci à ma petite amie Capucine Dalby, mes parents Véronique et Jean-François Risch et mes amis proches pour avoir su me remettre d'aplomb dans les moments compliqués.

Merci à l'école doctorale Sciences, technologies, santé de Reims. Aussi, je remercie particulièrement les rapporteurs Arnaud Martin et Gilles Kassel pour leur lecture attentive et les membres du jury Hacène Fouchal et Colette Faucher pour leur participation à la soutenance.

Enfin, je remercie mes directeurs de recherche Francis Rousseaux et Eddie Soulier.

English Preview

Title:

Improving text-classification models using the bag-of-concept paradigm

Abstract:

Most of text-classification methods use the "bag of words" paradigm to represent texts. However Bloahdom and Hortho have identified four limits to this representation: (1) some words are polysemics, (2) others can be synonyms and yet differentiated in the analysis, (3) some words are strongly semantically linked without being taken into account in the representation as such and (4) certain words lose their meaning if they are extracted from their nominal group. To overcome these problems, some methods no longer represent texts with words but with concepts extracted from a domain ontology (Bag of Concept), integrating the notion of meaning into the model. Models integrating the bag of concepts remain less used because of the unsatisfactory results, thus several methods have been proposed to enrich text features using new concepts extracted from knowledge bases. My work follows these approaches by proposing a model-enrichment step using a domain ontology, I proposed two measures to estimate to belong to the categories of these new concepts. Using the naive Bayes classifier algorithm, I tested and compared my contributions on the Ohsumed corpus using the domain ontology "Disease Ontology". The satisfactory results led me to analyse more precisely the role of semantic relations in the enrichment step. These new works have been the subject of a second experiment in which we evaluate the contributions of the hierarchical relations of hypernymy and hyponymy.

Table des Matières

Résumé **6**

Remerciements **8**

English **10**

Liste des Figures **16**

1 Introduction **20**
1.1 Historique . . . 20
1.1.1 L'apprentissage Automatique et la Classification de Textes . . . 20
1.1.2 Toujours Plus de Données . . . 22
1.2 Classification de Textes Représentés par des Concepts . . . 23
1.3 Présentation des Travaux . . . 24
1.4 Plan du Document . . . 24

2 État de l'Art **26**
2.1 Ontologies . . . 26
2.1.1 Définition . . . 26
2.1.2 Implémentation des Ontologies . . . 29
2.2 Classification Automatique de Textes . . . 31
2.2.1 Un Sous Domaine de l'Apprentissage Automatique . . . 31
2.2.2 Fonctionnement Général de la Classification de Textes . . . 33
2.2.3 Prétraitement de texte . . . 33
2.2.3.1 Extraction des Caractéristiques . . . 34
2.2.3.2 Sélection des Caractéristiques . . . 35
2.2.3.3 Transformation des Caractéristiques . . . 37
2.2.4 Représentation des Textes . . . 38
2.2.4.1 Représentation en Sac de Mots . . . 39
2.2.4.2 Représentation en Sac de N-Grammes . . . 40
2.2.4.3 Représentation en Sac de Groupes de Mots . . . 41
2.2.4.4 Représentation en Sac de Concepts . . . 41
2.2.4.5 Représentation Hybride . . . 42
2.2.4.6 Amélioration des Représentations . . . 42

2.2.5 Algorithmes d'Apprentissage Supervisé . . . 43
2.2.5.1 Les Méthodes Transductives et Inductives . . . 43
2.2.5.2 Machine à vecteurs de support . . . 44
2.2.5.3 Réseau de Neurones . . . 46
2.2.5.4 Arbre de décision . . . 48
2.2.5.5 Forêt d'Arbres Décisionnels . . . 50
2.2.5.6 Classifieur Naïf Bayésien . . . 52
2.2.6 Évaluation du Modèle de Classification . . . 53
2.2.6.1 Validation holdout . . . 54
2.2.6.2 Validation Croisée . . . 54
2.2.6.3 Mesure d'Évaluation . . . 55
2.2.7 Corpus de Textes . . . 56
2.3 Mégadonnées . . . 58
2.3.1 Définition en 5V . . . 58
2.3.1.1 Volume . . . 59
2.3.1.2 Variété . . . 59
2.3.1.3 Vitesse . . . 59
2.3.1.4 Véracité et Valeur . . . 60
2.3.2 Le Stockage de Données . . . 61
2.3.2.1 Définition . . . 61
2.3.2.2 Histoire des Bases de Données . . . 61
2.3.2.3 Base de Données Relationnelle . . . 63
2.3.2.4 Base de Données NoSQL . . . 64
2.3.3 Traitement des Mégadonnées . . . 65
2.3.3.1 le Système de Fichiers Distribués . . . 66
2.3.3.2 Framework de Calculs Distribués . . . 67
2.3.3.3 Hadoop . . . 68
2.3.3.4 Spark . . . 69
2.3.3.5 Autres Frameworks . . . 70

3 Classification de Textes Représentés par des Concepts 72
3.1 Introduction . . . 72
3.2 Ensembles de Données . . . 73
3.2.1 L'Ontologie Disease Ontology . . . 73
3.2.2 Corpus de Textes Ohsumed . . . 74
3.3 Représentation des Textes . . . 80
3.3.1 Extraction des Mots et des Groupes de Mots . . . 82
3.3.2 Association des Concepts . . . 86
3.4 Classification Naïve Bayésienne . . . 88
3.4.1 Apprentissage . . . 89
3.4.2 Classification . . . 91
3.5 Expérience . . . 91
3.5.1 Présentation des Modèles Testés . . . 91
3.5.1.1 Modèle Utilisant le Paradigme BoW . . . 92
3.5.1.2 Modèle Utilisant le Paradigme BoC . . . 93
3.5.2 Résultats . . . 93
3.6 Visualisation du Modèle . . . 94

3.6.1 Représentation du Modèle d'Apprentissage au Travers du Graphe de l'Ontologie . . . 95
3.6.2 Conception technique . . . 98

4 Enrichissement du Modèle de Classification de Textes 100
4.1 Introduction . . . 100
4.2 Méthode pour l'Enrichissement du Modèle . . . 101
4.2.1 Principe Général . . . 102
4.2.1.1 Présentation du Pseudo-Algorithme . . . 102
4.2.1.2 Explication par l'Exemple . . . 103
4.2.2 Mesure pour l'Enrichissement du Modèle . . . 105
4.2.3 Ajustement de la Mesure en Fonction de la Distance . . . 106
4.3 Expérience . . . 108
4.3.1 Présentation des Modèles Testés . . . 108
4.3.2 Résultats . . . 109
4.3.3 Critique des Résultats . . . 112
4.4 Résumé des Contributions . . . 113

5 Rôle des Relations Sémantiques dans L'Enrichissement de Modèle de Classification 114
5.1 Introduction . . . 114
5.2 Analyse par Type de Relation Sémantique . . . 115
5.2.1 Contexte des Recherches . . . 115
5.2.2 Choix et Analyse des Types de Relations Sémantiques . . . 116
5.2.3 Adaptation de l'Algorithme d'Enrichissement . . . 117
5.3 Expérience . . . 118
5.3.1 Présentation des Modèles Testés . . . 119
5.3.2 Résultats . . . 119
5.3.3 Critique des Résultats . . . 120
5.4 Résumé des Contributions . . . 122

6 Amélioration des Performances et Conception Technique 124
6.1 Amélioration des Performances . . . 125
6.1.1 Identification Des Temps d'Exécution . . . 125
6.1.2 Distribution des Traitements . . . 126
6.2 Conception Technique . . . 128
6.2.1 Langage et Environnement de Développement . . . 128
6.2.2 Librairies Externes . . . 129
6.2.3 Présentation des Méthodes Principales . . . 130
6.2.3.1 Extraction des Groupes Nominaux . . . 130
6.2.3.2 Création du Modèle . . . 131
6.2.3.3 Classification d'un Nouveau Texte . . . 131
6.2.3.4 Enrichissement du Modèle . . . 132

7 Conclusion et Perspective 134
7.1 Conclusion . . . 134
7.2 Perspectives . . . 135

Bibliographie **138**

Liste des Figures

2.1 Échantillon de l'ontologie *Disease Ontology*. 27
2.2 Différents types d'ontologies d'après Guarino. Les flèches liant les catégories d'ontologie indiquent la spécificité de ces dernières entre elles. . . . 28
2.3 Exemple de triplets RDF. 30
2.4 Exemple d'un graphe RDF. 30
2.5 Différence entre la classification supervisée et le clustering. 32
2.6 Étapes d'une classification de textes. 33
2.7 Principales techniques d'extraction des caractéristiques. 34
2.8 Table de contingence entre une caractéristique t et une catégorie C_i. . . . 36
2.9 États des textes lors d'une classification. 39
2.10 Principe général d'une méthode inductive. 44
2.11 Changement d'espace pour obtenir une séparation linéaire. 45
2.12 Marge optimale (la plus large) en vert et marge correcte mais non optimale en rouge. 46
2.13 Réseau de neurones simplifié à deux sorties (classification). 47
2.14 Principe de fonctionnement divisé en deux parties d'un neurone. 48
2.15 Données d'apprentissage pour l'arbre de décision. 49
2.16 Arbre de décision possible construit à partir des données d'apprentissage. 49
2.17 Prédiction d'un nouvel individu. 50
2.18 Forêts de trois arbres décisionnels avec sélection des variables et des individus. 51
2.19 Validation d'un modèle de classification supervisée. 54
2.20 Informations obtenues par une table de confusion. 55
2.21 Photo de deux IBM 350.Source : U. S. Army Red River Arsenal. 62
2.22 Exemple du wordCount avec la méthode de MapReduce 68

3.1 Définition du concept $DOID3077$ de DO. 75
3.2 Définition du concept $DOID3078$ de DO. 75
3.3 Contenu du fichier 0000682 de Ohsumed. 78
3.4 Répartition des textes de $Ohsumed - O$ et $Ohsumed - D$. 80
3.5 Graphique en barres de la répartition des textes de $Ohsumed - O$ et $Ohsumed - D$. 81
3.6 Graphique en barres de la répartition des textes de $Ohsumed - D$ et $Ohsumed - D\&C$. 81
3.7 Répartition des textes de $Ohsumed - D$ et $Ohsumed - D\&C$. 82
3.8 Workflow des techniques d'extraction des groupes nominaux. 83
3.9 Contenu du texte 0000014 de Ohsumed. 83
3.10 Étape de découpage du texte en phrases indépendantes. 84

3.11 Étape de découpage des phrases en unités lexicales. 84
3.12 Étiquetage morpho-syntaxique des unités lexicales. 85
3.13 Lémmatisation des unités lexicales 87
3.14 Extraction des groupes nominaux. 88
3.15 Recherche des concepts correspondant aux groupes nominaux contenus dans le texte 0000014. 89
3.16 Extrait du modèle d'apprentissage 90
3.17 Liste des traitements nécessaires à la création du Bag-of-Words. 92
3.18 Comparaison des modèles M_{BoW} et M_0 à l'aide du taux d'erreur, de la précision, du rappel et du F1-Score. 93
3.19 Visualisation des probabilités du concept *liver cirrhosis*. 95
3.20 Visualisation d'une partie du modèle d'apprentissage. 96
3.21 Visualisation d'une partie du modèle d'apprentissage avec l'ajout des concepts de l'ontologie non discriminants 97

4.1 Nombre de concepts du modèle après enrichissement en fonction de la distance d_{max}. 102
4.2 Exemple 1. Déroulé des étapes 0, 1 et 2 de l'algorithme. 104
4.3 Estimation des probabilités des concepts 4 et 5 pour les catégories C_1, C_2, C_3, et C_4. 105
4.4 Estimation des probabilités des concepts 2, 3 et 6 pour les catégories C_1, C_2, C_3, et C_4 et à l'aide de la $Mesure_1$. 106
4.5 Estimation des probabilités des concepts 2, 3 et 6 pour les catégories C_1, C_2, C_3, et C_4 et à l'aide de la $Mesure_2$. 107
4.6 Exemple 2. Modèle contenant 4 concepts de V liés à un concept potentiel. 107
4.7 Estimation des probabilités du concept 1 de l'exemple 2 en fonction de d_{max} et des mesures $Mesure_1$ et $Mesure_2$. 107
4.8 Évaluation des modèles M_0, M_1 et M_2. 109
4.9 Histogramme des résultats généraux de l'expérience. 110
4.10 Extrait du graphe du modèle $M_1(d_{max} = 2)$. Les nœuds oranges sont des concepts non inclus dans le modèle. 111

5.1 Description de la classe du concept *DOID_0050242*. 116
5.2 Cardinal de V en fonction des types de relation sémantique. 117
5.3 Exemple de l'étape d'enrichissement du modèle en fonction du type de relation sémantique. 118
5.4 Evaluation des modèles M_0, M_1, M_3 et M_4. 120
5.5 Exemple de l'étape d'enrichissement du modèle en fonction du type de relation sémantique. 121
5.6 Extrait des graphes des modèle $M_4(d_{max} = 2)$ (à gauche) et $M_3(d_{max} = 2)$ (à droite). Les nœuds oranges sont des concepts non inclus dans les modèles. 122

6.1 Temps d'exécution (TE) par mode de calcul. 126
6.2 Comparaison des performances en temps d'exécution des modes distribué et séquentiel 127

À Valentin Mattiroli, Stéphane Risch, Marie Lausch et Mathias Dymarski.

Chapitre 1

Introduction

L'analyse de données est un vieux domaine de recherche qui a pris un virage à 90 degrés lors de l'arrivée des premiers ordinateurs qui ont permis d'automatiser les traitements. A partir de cette évolution, de nouveaux axes de recherche sont apparus (dont l'intelligence artificielle moderne) entraînant avec eux des intérêts industriels. La première partie de cette introduction discute de l'historique liée à l'analyse automatique des données : de Turing dans les années 1950 aux traitements des données massives. La seconde partie précise le sujet de la classification de textes et des paradigmes de représentation des textes. La troisième partie expose la problématique de mon sujet de thèse et introduit mes contributions puis dans la quatrième et dernière partie, je présente le plan de la thèse.

1.1 Historique

1.1.1 L'apprentissage Automatique et la Classification de Textes

En 1950, Turing publiait son célèbre test d'intelligence artificielle (Turing, 1950). Un ordinateur passait le test si, lors d'une conversation à l'aveugle avec un humain, ce dernier ne se rendait pas compte qu'il échangeait avec un ordinateur. Ce test est considéré comme l'une des briques de base de l'intelligence artificielle moderne. Aussi, comme le rapporte (Guyon and Elisseeff, 2006), c'est durant ces années que Samuel employait pour la première fois le terme *machine learning*, traduit en français par la suite par *apprentissage automatique*. L'apprentissage automatique est la branche de l'intelligence artificielle ayant pour objectif le traitement automatique, par la machine, des données en vue de généraliser des comportements, on parle d'apprendre des modèles. Dans les années 1960, ce domaine connaissait un certain succès et c'est durant cette période que

de nombreux algorithmes ont été créés. Certains sont encore les bases des recherches actuelles, comme par exemple la méthode des k plus proches voisins inventée par Cover (Cover and Hart, 1967) dès 1967.

Pendant les trente années qui ont suivies, l'apprentissage automatique s'est développé sur de nombreuses applications : de la classification de données numériques (les revenus de tel individu lui permettent-ils de recevoir un prêt bancaire ?), textuelles (quel est le sujet principal de ce texte ?), sonores (quel est le morceau de musique se rapprochant le plus de celui-ci ?), images (quels sont les objets présents sur cette photo ?) ou encore vidéo (telle personne recherchée est-elle passée devant la caméra de surveillance ?). L'apprentissage automatique se divise en différents sous domaines dont les principaux sont la classification non supervisée aussi appelée clustering (regroupement de données sans catégorie pré-déterminée), la classification supervisée (généralisation de modèle à partir de données étiquetées), la classification semi-supervisée (utilisation de données incomplètes), la classification par renforcement, etc.

La classification de textes appartient au domaine de la classification supervisée et a pour objectif la généralisation de données textuelles pré-étiquetées afin de prédire la catégorie (classe, sujet, thème) de nouveaux textes. Ce domaine de recherche est né de la motivation d'automatiser un processus initialement manuel en s'appuyant sur des méthodes d'apprentissage automatique, de traitement automatique du langage (TAL) et de recherche d'information (IR). Les applications sont diverses : allant de la détection de spams, à l'identification automatique de la langue en passant par l'analyse de sentiments. Malheureusement, il n'existe pas une méthode unique permettant de résoudre tous les problèmes, bien au contraire, la mise en place d'une application de classification de textes implique le choix du paradigme de représentation des textes (un texte peut être représenté par une liste de mots, de groupes nominaux, de concepts, etc.) mais aussi des méthodes nécessaires à l'extraction des caractéristiques des textes (certaines sont statistiques, d'autres proviennent du TAL) et enfin de l'algorithme d'apprentissage.

Malgré un certain intérêt des industriels à l'égard des applications d'apprentissage automatique (et notamment de la classification de textes), celui-ci a subit un déclin important dans les années 1980. En effet, les généralisations des données n'étaient pas satisfaisantes et ne permettaient pas aux utilisateurs d'y attacher un degré de confiance suffisant. Cependant, un facteur soudain allait relancer la machine : l'explosion de la production et du stockage de données promettant une meilleure qualité des modèles et donc de meilleures prédictions.

1.1.2 Toujours Plus de Données

Dans les années 1960, deux ordinateurs ont communiqué pour la première fois au travers du réseau Internet. Dans les années 1990, c'est le WEB qui fait son émergence facilitant l'accès à Internet aux nouveaux utilisateurs. Ces dernières années, nous assistons à l'arrivée des objets connectés, introduisant l'objet comme acteur du réseau. Ces petites révolutions technologiques engendrent la création et la consommation de données toujours plus nombreuses, diverses et complexes si bien que la communauté leur a donné un nom : les données massives (Big Data). Pendant des années les entreprises ont tellement accumulé et conservé les données que les systèmes informatiques les plus performants n'étaient plus capables de les traiter dans un temps raisonnable. Pourtant, l'intérêt de celles-ci était déjà présent : tant de données permettraient de résoudre les problèmes de qualité des modèles d'apprentissage automatique par une meilleure généralisation.

La recherche s'est longtemps satisfaite des améliorations des performances des machines en termes de nombre de traitements par seconde. Cependant, les performances des composants informatiques (dont principalement le micro-processeur) n'arrivent plus à suivre la courbe exponentielle de l'évolution des données. Pourtant une entreprise avait trouvé la solution depuis longtemps.

En 2003, Google[1] publiait (Ghemawat et al., 2003), cet article présente le système de gestion des fichiers de l'entreprise. Cinq ans plus tard, Google publie deux nouveaux papiers de recherche présentant leurs méthodes de calculs capables de traiter une grande quantité de données en un temps restreint. Le premier article (Dean and Ghemawat, 2008) traite de la distribution des calculs au sein d'un cluster informatique via le patron d'architecture MapReduce. Le second (Chang et al., 2008) présente le système de gestion de base de données (SGBD) NoSQL orienté colonnes permettant le stockage de données dans un environnement distribué. Ces publications ont permis l'émergence d'outils libres dont les plus connus sont Apache Hadoop (White, 2012) et Apache Spark (Zaharia et al., 2012). Ceux-ci répondent à la problématique posée par les données massives en réduisant drastiquement les temps de calculs de celles-ci. Pourtant, cette solution introduit une contrainte forte : la distribution des calculs et donc l'indépendance des données et des calculs entre eux. Cependant la plupart des algorithmes d'apprentissage automatique sont séquentiels, les rendant inutilisables dans des environnements distribués. Des initiatives récentes ont mené à la création de nouveaux algorithmes adaptés ou à la mise à jour, non sans difficulté, d'anciens algorithmes séquentiels. (Koturwar et al., 2015) présente un panorama de ces méthodes.

[1] https://www.google.fr/

L'intégration de la contrainte liée à la gestion des Big Data est essentielle pour les entreprises de services comme Capgemini. Celle-ci assure la réutilisation des programmes et facilite leur déploiement, c'est pour cette raison que l'intégration de mes travaux dans un environnement Big Data s'est imposé.

1.2 Classification de Textes Représentés par des Concepts

L'arrivée du Big Data et les solutions de traitement associées ont permis de remettre l'apprentissage automatique dont la classification de textes sur le devant de la scène, si bien que nombre de projets industriels en dépendent. Les applications de maintenance prédictive, d'annotation et classement d'articles journalistiques et de prédiction de diagnostics médicaux sont des exemples de cas d'utilisation.

Les principales méthodes de ce domaine sont décrites dans (Aggarwal and Zhai, 2012), (Korde and Mahender, 2012), (Harish et al., 2010) et (Aas and Eikvil, 1999). La grande majorité de ces dernières base la représentation des textes sur le paradigme du bag-of-words (BOW) où chaque mot de ces derniers est associé à une mesure telle que la fréquence d'apparition dans le dit texte. (Bloehdorn and Hotho, 2004) a identifié quatre limites à cette représentation : certains mots sont polysémiques ("orange" se réfère à la couleur, le fruit, l'opérateur téléphonique ou encore la ville du sud de la France), d'autres peuvent être des synonymes et être malgré tout différenciés dans l'analyse ("plectre" a le même sens que "médiator"), d'autres encore sont fortement liés sémantiquement sans que cela soit pris en compte dans la représentation ("chevalet" est très lié à "frette") et enfin, certains mots perdent leur sens s'ils sont extraits de leur groupe nominal (exemple : "poste de police" n'a pas le même sens que "poste" et "police" pris séparément). A ces limites s'ajoute celle du nombre trop importants de mots représentant les documents qu'on appelle la malédiction de la dimension. (Beyer et al., 1999) a montré qu'à partir d'un nombre suffisamment important de dimensions, les observations (des textes dans notre étude) deviennent équidistantes et sont donc difficilement comparables.

Une alternative au sac de mots et le sac de concepts (BOC) (Sahlgren and Cöster, 2004) où chaque texte est représenté non plus par une collection de mots mais par des concepts représentant un sens. Il peut s'agir de concepts extraits d'ontologie (Kehagias et al., 2003) ou d'autres sources comme wikipedia[2] (Huang et al., 2009). En utilisant cette représentation des textes, (Kehagias et al., 2003) a obtenu de mauvais résultats, cependant (Bloehdorn and Hotho, 2004) a montré qu'avec une approche hybride (les textes sont représentés par des mots et des concepts) la méthode apportait des résultats plus satisfaisants. Néanmoins, l'hybridité de ces travaux est, malgré les bons résultats,

[2] http://www.wikipedia.fr/

une source d'erreurs car des mots bruts composent la représentation des textes et sont donc toujours ininterprétables par la machine.

1.3 Présentation des Travaux

La classification de textes impose que les textes soient représentés, suivant le paradigme utilisé, par les mots, groupes de mots ou concepts qu'ils contiennent. Différentes méthodes permettent d'enrichir ces représentations : la première a pour objectif de remplacer les mots et groupes de mots par des concepts d'une ontologie (Hotho et al., 2003), la seconde consiste à ajouter les concepts contenus dans les textes à leur représentation (Benkhalifa et al., 2001), enfin une troisième méthode vise à ajouter des concepts non contenus dans les textes à l'aide d'une base de connaissance telle qu'une ontologie, WordNet (Fellbaum, 1998) ou des sites WEB comme Wikipédia[3] (Wang et al., 2007). Cette troisième méthode a l'avantage d'ajouter de nouvelles connaissances externes à la représentation des textes et donc au modèle, cependant cette intégration est réalisée au niveau de la représentation des textes empêchant de s'appuyer sur les règles du modèle pour le calcul des mesures des nouveaux concepts.

En m'inspirant de ces travaux, j'ai proposé la mise en place d'une technique permettant d'enrichir non pas les caractéristiques des textes mais le modèle à part entière. Cette solution offre de calculer les mesures des nouveaux concepts à partir de l'ensemble des concepts du modèle, optimisant ainsi leur généralisation.

Afin d'évaluer et de comparer mes travaux, j'ai fait le choix d'utiliser le classifieur naïf Bayésien (Lewis, 1998) pour sa large utilisation dans la communauté et sa simplicité. Aussi, l'ensemble des expériences menées sont réalisées à l'aide du corpus Ohsumed (Hersh et al., 1994) et de l'ontologie de domaine Disease Ontology (DO) (Schriml et al., 2012).

1.4 Plan du Document

Dans un premier chapitre, je présente un état de l'art complet des domaines qui ont impacté mes recherches. Je commence en définissant les ontologies informatiques et en présentant leurs possibilités d'implantation puis je continue en décrivant les méthodes, outils et corpus de l'apprentissage automatique de textes, enfin je termine en fournissant un panorama du sujet des Big Data. Le chapitre suivant introduit le sujet de la classification de textes représentés par des concepts. Dans celui-ci, je propose un modèle sur

[3] http://wikipedia.com/

lequel je me baserai pour évaluer mes contributions. Le quatrième chapitre expose mes premières contributions : je propose deux mesures pour l'enrichissement du modèle de classification de textes représentés par des concepts à l'aide d'une ontologie de domaine. Les modèles enrichis sont évalués et comparés avec le modèle de base présenté dans le chapitre 3. Dans le chapitre suivant, je propose une analyse approfondie de la méthode d'enrichissement exposée dans le chapitre 4. J'y discute et évalue le rôle des relations sémantiques dans l'enrichissement du modèle. Le sixième chapitre présente les choix de conception technique de l'application. J'y justifie l'utilisation d'un environnement Big Data et présente mes contributions techniques d'une part et celles empruntées de l'autre. Enfin, je conclus et présente les perspectives dans un septième chapitre.

Chapitre 2

État de l'Art

Mes travaux s'inscrivent dans 3 grands domaines de recherche : les ontologies (au sens informatique), l'apprentissage automatique (et plus spécifiquement la classification supervisée de textes) et enfin le Big Data. La section état de l'art présente ces trois domaines distinctement.

2.1 Ontologies

2.1.1 Définition

Suivant le domaine de compétence, une ontologie peut avoir différentes définitions. Ainsi, les domaines de la philosophie et de l'informatique ont chacun une définition du terme. Cette double définition peut engendrer des incompréhensions suivant le domaine de compétence du lecteur. (Guarino, 1998) formalise cette dernière et discute de la distinction entre la définition philosophique et la définition informatique. Ainsi, pour éviter la confusion entre ces deux définitions, l'ontologie philosophique est appelée une conceptualisation et l'ontologie informatique est désignée par le terme de base : une ontologie. Dans ce document, le terme *ontologie* se rapporte donc à une ontologie informatique. Dans (Gruber, 1993), Gruber définit une ontologie de la manière suivante : *une spécification d'un vocabulaire représentatif d'un domaine partagé de discours - définitions de classes, de relations, de fonctions et d'autres objets - s'appelle une ontologie*[1]. Dans (Gruber, 1995), il simplifie (et élargit) sa définition par : *une ontologie est une spécification explicite d'une conceptualisation*[2]. Ses définitions montrent le lien existant entre

[1] La citation dans la langue originale est *A specification of a representational vocabulary for a shared domain of discourse — definitions of classes, relations, functions, and other objects — is called an ontology*

[2] La citation dans la langue originale est *An ontology is an explicit specification of a conceptualization.*



FIGURE 2.1: Échantillon de l'ontologie *Disease Ontology*.

les visions philosophique et informatique. Une conceptualisation représente ainsi une certaine vision du monde alors qu'une ontologie est une représentation de cette vision. Comme Gruber l'indique dans sa définition, cette représentation est une spécification. Cela signifie qu'elle est décrite de manière formelle dans le but (notamment) de pouvoir être traitée par des outils informatiques. Ainsi, une ontologie est une spécification d'une conceptualisation dans le sens où toutes les connaissances de la conceptualisation ne sont pas représentables informatiquement[3].

Plus concrètement, une ontologie correspond à un ensemble de concepts liés par des relations de subsomption et des relations sémantiques. Une ontologie est souvent représentée par un graphe où les nœuds sont les concepts et les arcs les relations. La figure 2.1 représente un échantillon de l'ontologie médicale *Disease Ontology* produite par l'Université du Maryland[4].

Il existe différents niveaux d'ontologie. (Guarino, 1997) propose trois niveaux d'ontologie résumés par la figure 2.2.

[3]Par exemple, les sensations ressenties par un Homme peuvent être incluses dans une conceptualisation alors qu'il est très compliqué de les représenter dans une ontologie.

[4]http://disease-ontology.org/

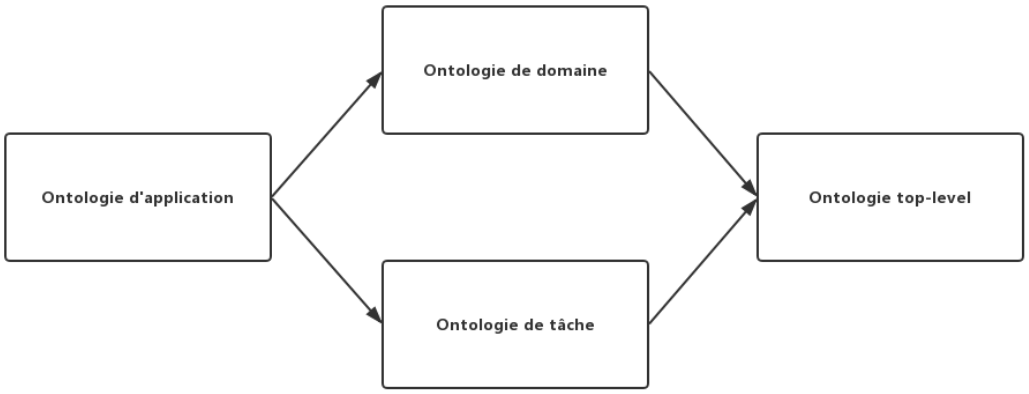


FIGURE 2.2: Différents types d'ontologies d'après Guarino. Les flèches liant les catégories d'ontologie indiquent la spécificité de ces dernières entre elles.

- Ontologie globale (Top-level ontology). Une ontologie globale est une ontologie décrivant des concepts très généraux indépendants de tout contexte. Guarino donne les exemples de la description des concepts du temps, de la matière, d'un événement, etc.

- Ontologie de domaine (Domain ontology) et ontologie de tâche (Task ontology). Les ontologies de domaine et de tâche sont considérées comme étant au même niveau. Chacune d'elles a pour objectif de décrire le vocabulaire du domaine (par exemple : l'aéronautique, les maladies humaines, etc.) ou de la tâche (par exemple : la vente).

- Ontologie d'application (Application ontology). Une ontologie d'application décrit un champ d'application dépendant d'une part d'une ontologie de domaine et de l'autre d'une ontologie de tâche (par exemple : maintenance des moteurs des avions Airbus A380).

Dans le but de définir les règles de l'ontologie informatique, Gruber a exposé 5 critères décrits ci-après (Gruber, 1995) :

- Clarté. Une ontologie doit partager précisément le sens du terme dans le but d'être le plus objectif possible. La définition de chaque composant d'une ontologie doit être le plus indépendant possible de tout contexte social. Gruber ajoute que si une définition peut être établie à partir d'axiomes, alors elle doit l'être. Aussi, chaque définition doit être documentée en langage naturel.

- Cohérence. Une ontologie doit être cohérente en le sens ou les définitions des concepts ne doivent pas entrer en conflit. Également, la cohérence doit être maintenue dans la documentation écrite en langage naturel.
- Extensibilité. Une ontologie doit pouvoir anticiper une série de tâches telle que l'utilisation du vocabulaire partagé. Une ontologie doit pourvoir être étendue sans que cela ne nécessite la modification de définitions préexistante.
- Déformation d'encodage minimal. Une ontologie a pour vocation d'être encodée et décrite par un langage informatique. Encoder des concepts peut poser problème et causer des biais. Gruber insiste en affirmant que les choix de représentation d'une ontologie ne doivent pas dépendre de l'implémentation.
- Engagement ontologique minimal. Une ontologie doit définir le plus petit vocabulaire possible permettant de décrire un domaine.

Les ontologies ont pour vocation d'être interprétables par des machines. Pour ce faire, différents outils ont été développés par les chercheurs mais aussi les industriels. La partie suivante présente les principaux langages et logiciels du moment.

2.1.2 Implémentation des Ontologies

Comme Gruber le sous-entend dans la définition de ses règles, une ontologie doit pouvoir être interprétée et étendue par un ensemble de machines. Ainsi, l'ontologie doit être encodée de manière à ce que des machines ne communiquant pas puissent la décoder. L'encodage d'une ontologie suit des règles strictes dictées par un langage de spécification. Plusieurs langages ont été mis en place pour décrire les ontologies, mais ce sont ceux du web sémantique qui dominent, grâce notamment à l'effort de l'organisme de standardisation du W3C. (Su and Ilebrekke, 2002) présente une étude comparative des langages de spécification. Actuellement, le langage recommandé pour décrire les ontologies est le langage OWL (Ontology Web Language). Ce dernier est fondé sur le langage RDF Schéma (RDFS) lui-même fondé sur le langage RDF (Resource Description Framework). Ci-après je décris en détails ces trois langages :

- RDF (Klyne and Carroll, 2006). RDF est le langage sur lequel le web sémantique est fondé dont l'une des syntaxes la plus connue et utilisée est le RDF/XML. Ce langage a pour vocation de décrire des ressources web identifiées par une URL, une URI et plus récemment une IRI.

  Le RDF est fondé sur la notion de triplet. Un triplet est l'association d'un sujet et d'un objet via un prédicat. Un triplet peut être vu comme une relation liant

Sujet	Prédicat	Objet
http://ex.fr/ People/KevinParker	http://ex.fr/ pred/chanteur	http://ex.fr/ Band/TameImpala
http://ex.fr/ People/KevinParker	http://ex.fr/ pred/producteur	http://ex.fr/ Band/Pond
http://ex.fr/ People/KevinParker	http://ex.fr/pred/nom	Kevin Parker

FIGURE 2.3: Exemple de triplets RDF.

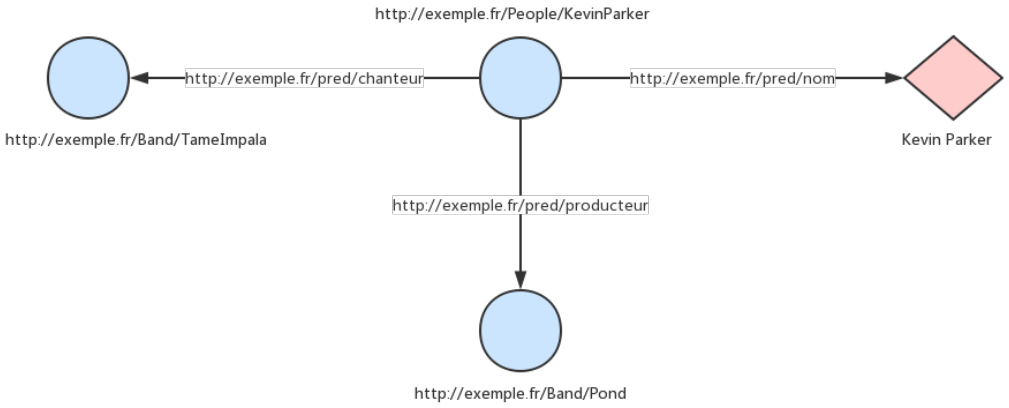


FIGURE 2.4: Exemple d'un graphe RDF.

deux nœuds d'un graphe. Le sujet est le nœud de départ, la relation correspond au prédicat et l'objet au nœud de destination. Cette notion de base explique pourquoi RDF est vu comme un modèle de graphe. Contrairement au prédicat qui est nécessairement identifié par une URI, le sujet et l'objet peuvent être des nœuds anonymes (non identifié par une URI). En guise d'exemple, la figure 2.4 représente le graphe des triplets RDF du tableau 2.3.

- RDFS (Brickley and Guha, 2000). RDFS est un langage extensible fondé sur le RDF dont la première recommandation a été publiée dès 2004 par le W3C. RDFS propose les briques de bases permettant de représenter les ontologies à partir du langage RDF tels que $rdfs : Class$, $rdfs : domain$, $rdfs : range$, $rdfs : label$ ou encore $rdfs : SubClassOf$.

- OWL (McGuinness and Van Harmelen, 2004). OWL est un langage extensible fondé sur RDFS (et donc par extension sur RDF) et ayant pour objectif de définir les ontologies. OWL est plus expressif que RDFS, c'est pour cette raison qu'il est devenu une recommandation du W3C en 2012. OWL est le langage le plus

largement utilisé par la communauté et est défini par trois sous langages : OWL-Lite, OWL-DL et OWL-Full. Chaque sous-langage est plus ou moins expressif (OWL-Lite étant le moins expressif et OWL-Full le plus expressif).

Afin d'interroger les ontologies écrites en RDF, RDFS ou OWL, la communauté a mis en place plusieurs langages de requête dont le plus répandu est SPARQL (SPARQL Protocol and RDF Query Language) (Prud'Hommeaux and Seaborne, 2008). Cette popularité est due entre autre à sa recommandation dès 2008 par le W3C. Ainsi SPARQL est l'équivalent de SQL pour les bases de données, ce langage permet la sélection, la modification, la suppression et la création de données au sein d'un graphe RDF.

La manipulation des ontologies par les différents langages de spécification et de requêtes reste une tâche complexe et nécessite une certaine expertise. De plus, les ontologies sont écrites en code ce qui les rend difficiles à représenter. Plusieurs logiciels ont été créés par la communauté dans le but de simplifier l'utilisation des ontologies et de les visualiser. (Kapoor and Sharma, 2010) et (Duineveld et al., 2000) présentent chacun une étude comparative d'éditeurs d'ontologie. Ce milieu évoluant très rapidement, la plupart des éditeurs présentés n'existent plus à l'exception de Protégé.

Protégé (Gennari et al., 2003) est l'éditeur d'ontologies le plus connu et le plus utilisé par la communauté avec plus de 310000 utilisateurs. Il a été créé à l'université de Stanford où il est toujours maintenu. Cet outil écrit en Java est extensible grâce à un système de plug-in. Enfin, Protégé suit les recommandations du W3C et supporte OWL 2 et RDF.

Outre ces outils, la communauté a également mis en place des librairies (APIs) ayant pour but de manipuler les ontologies au sein de programme. L'API la plus connue est JENA (McBride, 2001). Cette dernière permet la gestion des ontologies écrites en RDF et OWL. Elle permet également l'interrogation des ontologies à l'aide de SPARQL. JENA est disponible pour le développement en Java et est soutenu par la fondation Apache. OWL API (Horridge and Bechhofer, 2011) est une API Java dédiée à la manipulation des ontologies écrites à l'aide du langage de spécification OWL.

2.2 Classification Automatique de Textes

2.2.1 Un Sous Domaine de l'Apprentissage Automatique

L'apprentissage automatique (Machine Learning, ML) est un domaine de recherche à la croisée de l'intelligence artificielle, la statistique et l'informatique. Ce domaine a émergé dans les années 1960 en ayant pour objectif de fournir à la machine un ensemble

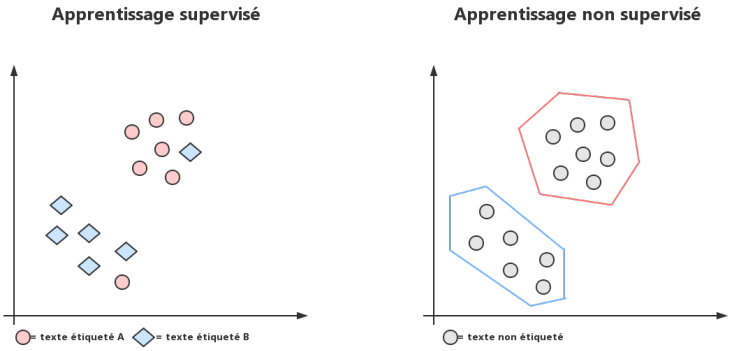


FIGURE 2.5: Différence entre la classification supervisée et le clustering.

d'algorithmes lui permettant d'apprendre des modèles statistiques à partir de données et d'informations, et ce, de manière automatique. De nombreux articles et livres proposent des panoramas de ce domaine. (Mjolsness and DeCoste, 2001) propose un rapide état de l'art sur le sujet et expose les défis à relever. (Cornuéjols and Miclet, 2011) et (Tufféry, 2012) sont les deux principales références de l'apprentissage automatique pour les francophones.

Il n'existe pas un algorithme capable de résoudre tous les problèmes de machine learning, ainsi le domaine est divisé en plusieurs sous domaines : l'apprentissage non-supervisé dont (Hastie et al., 2009), (Xu and Wunsch, 2005) et (Grira et al., 2004) présentent les principales techniques, l'apprentissage supervisé dont (Kotsiantis et al., 2007) propose un panorama, l'apprentissage par transfert dont (Pan and Yang, 2010) présente un état de l'art, l'apprentissage par renforcement dont les principales techniques ont été analysées par (Kaelbling et al., 1996) et encore l'apprentissage semi-supervisé revu par (Zhu, 2005). Suivant les données d'entrée d'une analyse, le data scientist s'orientera vers l'un des domaines cités ci-dessus. Par exemple, des données brutes non étiquetées et sans valeurs manquantes vont mener le data scientist vers les méthodes d'apprentissage non supervisé alors que des données étiquetées mènent vers l'apprentissage supervisé. Ces deux sous domaines de l'apprentissage automatique sont dominants actuellement. La figure 2.5 présente rapidement le principe de ces derniers.

L'apprentissage automatique a un vocabulaire spécifique que je vais utiliser dans ce document. On dit qu'on analyse des individus (ou observations) statistiques. L'ensemble des individus correspond à la population de l'analyse. Un individu est décrit par un vecteur regroupant n variables. Une variable décrit une caractéristique de l'individu, elle peut être de différents types : numérique, catégorielle, etc. Le nombre de variables correspond à la dimension du vecteur.

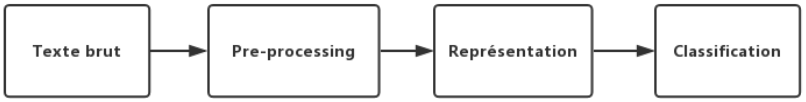


FIGURE 2.6: Étapes d'une classification de textes.

2.2.2 Fonctionnement Général de la Classification de Textes

La classification de textes est un sous-domaine particulier de l'apprentissage automatique utilisant des données d'entrée exclusivement textuelles. Tout comme l'apprentissage automatique, la classification de textes se divise en autant de sous domaines qu'il n'y a de types de données textuelles différents. Par exemple, un corpus de textes bruts pré-catégorisés permet la création de modèle d'apprentissage supervisé en vue de prédire la catégorie de nouveaux textes non étiquetés. Un autre exemple est le regroupement de textes sans étiquette. L'objectif est la création de groupes de textes homogènes (traitant d'un sujet commun) et hétérogènes entre eux (deux groupes de textes traitent de sujets différents). Ce dernier exemple est parfois nommé le clustering de textes le distinguant du premier (Sasaki and Shinnou, 2005).

Le déroulement d'une classification de textes suit quatre étapes principales résumées par la figure 2.6. La première est l'acquisition du texte brut. Un texte brut a la propriété de ne contenir aucune structure. La seconde est le prétraitement de ce dernier (pre-processing) ayant pour objectif de collecter les caractéristiques du texte (features). Une caractéristique est un mot, un groupe de mots ou encore un concept. La troisième étape est la représentation du texte. Il s'agit de présenter le texte comme un vecteur numérique interprétable par les algorithmes de classification. Enfin, la quatrième et dernière étape est le traitement, que ce soit par l'algorithme d'apprentissage ou celui de classification.

2.2.3 Prétraitement de texte

Le prétraitement de texte a pour objectif la représentation sous forme de vecteur de caractéristiques. Une caractéristique est un mot, groupe de mots ou concept qui décrit le texte. Suivant la représentation souhaitée, différentes étapes sont nécessaires à la construction du vecteur : l'extraction (feature extraction), la sélection (feature selection) et la transformation (feature transformation) des caractéristiques. Régulièrement des confusions sont faites entre les étapes d'extraction et de transformation. Certains considèrent que l'extraction concerne uniquement l'ensemble des techniques permettant d'extraire les mots à partir du texte brut (Zu et al., 2003). D'autres lui ajoutent l'idée

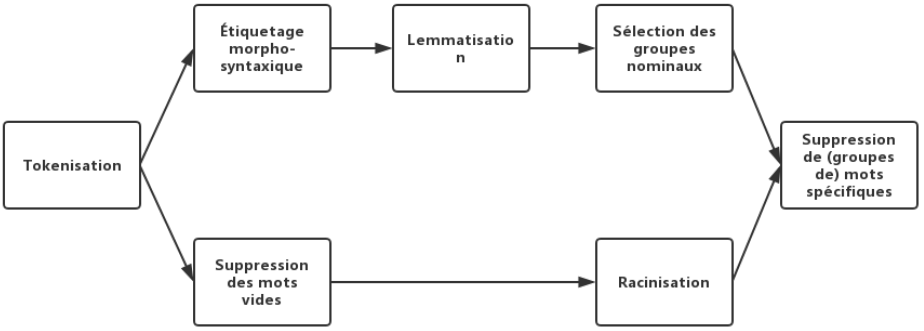


FIGURE 2.7: Principales techniques d'extraction des caractéristiques.

de transformation (Zareapoor and Seeja, 2015) : extraire de nouvelles caractéristiques à partir d'une liste de départ. Dans cet état de l'art, nous distinguons les deux ensembles de techniques. Les sous sections suivantes présentent l'intérêt de chacune des étapes de prétraitement de texte et les principales techniques utilisées en classification supervisée de textes.

2.2.3.1 Extraction des Caractéristiques

L'extraction des caractéristiques est une étape indispensable de la classification (Guyon and Elisseeff, 2006) et notamment de textes (Lewis, 1992a). Il n'est pas rare que celle-ci soit directement et exclusivement associée au processus de prétraitement. Cette étape regroupe les techniques d'extraction des caractéristiques à partir du texte brut. La figure 2.7 présente l'ordre d'utilisation générale des différentes techniques. Suivant le mode de représentation des textes choisi, certaines techniques sont dispensables, d'autres moins.

- Tokenisation : Cette étape a pour objectif la segmentation d'un texte en unités atomiques. On appelle ces unités les "tokens". Généralement on divise le texte en mots ou phrases. Par exemple la tokenisation en mots de la phrase "Une guitare possède six cordes" donne les tokens suivants : "Une", "guitare", "possède", "six" et "cordes".

- Étiquetage morpho-syntaxique (POS tagging) (Brill, 1995) : Cette étape vise à identifier et annoter les informations grammaticales de chacun des mots. Par exemple, il s'agit d'associer la fonction grammaticale "verbe" au mot "possède" de la phrase présentée ci-dessus.

- Suppression des mots vides (stop words) : Les mots vides sont des mots qui n'apportent pas (ou peu) d'information dans un texte. On dit qu'ils ne sont pas

significatifs. Par exemple, les déterminants tels que "le", "la", "du", les conjonctions de coordination telles que "donc", "car" sont des mots vides. Dans le cas de la représentation des textes avec des mots simples, ces mots vides sont supprimés. Dans le cas de la représentation avec des groupes de mots et/ou des concepts, ces mots sont nécessaires à la découverte de groupes nominaux et sont donc conservés.

- Racinisation (stemming) (Lovins, 1968) : La racinisation permet d'extraire le radical de chaque mots. Cette étape est essentielle dans la phase de prétraitement, notamment pour la réduction du nombre de caractéristiques d'un document. Par exemple, le radical des mots "chercher", "chercheurs" et "recherche" est "cherch".

- Lémmatisation : La lémmatisation a un objectif similaire à la racinisation : réduire le nombre de caractéristiques. Cependant, il ne s'agit pas d'extraire le radical des mots mais leur lemme en retournant les verbes à l'infinitif et les noms au masculin singulier. A la différence du radical, le lemme est un vrai mot du dictionnaire. Par exemple, le lemme du mot "chercheurs" est "chercheur".

- Sélection des mots et groupes de mots : Cette étape permet la sélection des mots et groupes de mots en fonction de leur fonction grammaticale. Elle permet également la réduction de la dimension. Dans le cas d'une correspondance avec les concepts d'une ontologie, il est usuel de ne sélectionner que les groupes nominaux.

- Suppression de mots et groupes de mots spécifiques : Généralement, le nombre des caractéristiques est élevé malgré l'élagage subit. Pour diminuer cette dimension, il est courant de supprimer les mots peu fréquents à partir d'un seuil (2 ou 3 mots minimum) ou de supprimer des mots ou groupe de mots directement à partir d'une liste noire.

L'extraction des caractéristiques permet de retourner des mots ou groupes de mots présents dans le document. Malgré les différentes techniques de l'extraction pour réduire la dimension, cette dernière peut rester suffisamment élevée pour avoir recours à d'autres techniques de réduction de la dimension.

2.2.3.2 Sélection des Caractéristiques

La sélection des caractéristiques a deux objectifs : diminuer le nombre de caractéristiques de l'analyse et supprimer certaines caractéristiques susceptibles de dégrader la qualité de la discrimination. Elle a été initiée par (Marill and Green, 1963) et regroupe désormais de nombreuses techniques. Dans son étude, (Yang and Pedersen, 1997) compare les cinq principales méthodes : DF, IG , MI, CHI et TS

	C_i	$NON(C_i)$	Total
t	A	B	A+B
$NON(t)$	C	D	C+D
Total	A+C	B+D	N

FIGURE 2.8: Table de contingence entre une caractéristique t et une catégorie C_i.

- Document Frequency (DF). Cette technique propose d'associer à chaque caractéristique le nombre de documents dans lesquels elle apparaît. A l'aide d'un seuil T fixé à l'avance, les caractéristiques peu fréquentes (se retrouvant dans moins de T documents) sont supprimées. L'hypothèse derrière cette méthode est que les termes peu fréquents n'apportent pas d'information et n'améliorent pas la discrimination. Malgré une apparente simplicité, cette méthode de sélection apporte de bons résultats.

- Information gain (IG). Cette mesure est principalement utilisée dans l'apprentissage des arbres de décision (Quinlan, 1986). L'objectif est d'associer à chaque caractéristique t une valeur de gain noté $G(t)$. Soit l'ensemble $C = \{C_1, C_2, C_k\}$ de k catégories, pour chaque caractéristique, le gain est calculé par :

$$\begin{aligned} G(t) = &- \sum_{i=1}^{k} P(C_i) \log P(C_i) \\ &+ P(t) \sum_{i=1}^{k} P(C_i|t) \log P(C_i|t) \\ &+ P(\bar{t}) \sum_{i=1}^{k} P(C_i|\bar{t}) \log P(C_i|\bar{t}) \end{aligned} \tag{2.1}$$

  De la même façon que la mesure DF, un seuil est mis en place et suivant la valeur de $G(t)$ la caractéristique est conservée ou exclue du vecteur.

- Information mutuelle (Mutual Information, MI). L'information mutuelle est une mesure déterminant la dépendance entre une caractéristique t et une catégorie C_i de telle manière à ce qu'une indépendance totale obtienne une mesure égale à 0.

  Le tableau 2.8 est la table de contingence résumant les informations suivantes. A correspond au nombre de fois où t et C_i co-occurrent, B le nombre de fois où t occurre en dehors de la catégorie C_i, C le nombre de fois où t n'est pas présent dans la catégorie C_i, D le nombre de fois où ni t ni C_i n'apparaissent et N le nombre total de textes. On note cette mesure $I(t, C_i)$ et elle est définie par :

$$I(t, C_i) = \log \frac{P(t \Lambda C_i)}{P(t) * P(C_i)} \tag{2.2}$$

L'information mutuelle est estimée par :

$$I(t, C_i) \approx \log \frac{A * N}{(A + C) * (A + B)} \tag{2.3}$$

- χ^2. La mesure du χ^2 permet de calculer la dépendance entre une caractéristique t et la catégorie C_i. Pour un couple donné, plus cette mesure a une forte valeur, plus la caractéristique et la catégorie sont dépendantes. En se basant sur la table de contingence 2.8 et la notation de l'information mutuelle, la mesure $\chi^2(t, C_i)$ du χ^2 s'obtient par :

$$\chi^2(t, C_i) = \frac{N * (AD - CB)^2}{(A + C) * (B + D) * (A + B) * (C + D)} \tag{2.4}$$

- Term Strength (TS). Cette méthode a été initialement proposée par Wilbur dans (Wilbur and Sirotkin, 1992) puis appliquée à la classification de textes dans (Yang, 1995) et (Yang and Wilbur, 1996). Cette mesure diffère fortement des quatre précédentes. L'hypothèse de cette dernière est la suivante : une caractéristique t est considérée comme importante si elle occurre dans de nombreux textes d'une même catégorie. Soient x et y deux textes distincts d'une même catégorie. Pour chaque caractéristique de l'analyse, on calcule la mesure $TS(t)$ de la manière suivante :

$$TS(t) = P(t \in y | t \in x) \tag{2.5}$$

Suivant un seuil de considération de relation entre deux textes, les caractéristiques sont conservées ou rejetées.

2.2.3.3 Transformation des Caractéristiques

L'étape de transformation des caractéristiques a pour objectif la modification et la recomposition de l'ensemble de ces dernières. Cette étape n'est pas indispensable pour la classification de textes mais peut être essentielle dans d'autres applications notamment pour la discrétisation ou encore la normalisation des données (Kusiak, 2001). (Zu et al., 2003) présente un court résumé des méthodes de feature transformation pour la classification de données textuelles.

- Normalisation de la fréquence des caractéristiques. Cette transformation a pour objectif le passage d'une fréquence d'occurrence absolue (et dépendante de la longueur

du texte) à une fréquence relative à l'ensemble des mots. Soit x_i la fréquence de la caractéristique i, on note y_i la fréquence relative :

$$y_i = \frac{x_i}{\sum_{j=1}^{n} x_j} \tag{2.6}$$

avec n le nombre total de caractéristiques.

- Power Transformation (PT). Cette transformation a pour objectif de stabiliser la variance simulant ainsi une distribution normale et améliorant la symétrie des caractéristiques. On note z_i la transformation de la caractéristique i :

$$\begin{aligned} &z_i = x_i^v \\ &avec(0 < v < 1) \end{aligned} \tag{2.7}$$

- Analyse en Composante Principale (ACP). L'analyse en composante principale (Wold et al., 1987), également appelée la transformation de Karhunen et Loeve (KLT) (Fukunaga and Koontz, 1970) en traitement de l'image est une méthode de transformation de caractéristiques corrélées en caractéristiques non corrélées permettant ainsi la réduction de la dimension. Cette technique est coûteuse en calcul, ce qui la rend peu utilisée dans le domaine de la classification de textes qui traite de nombreux textes. Une démonstration complète de la méthode est disponible via (Jolliffe, 2002).

2.2.4 Représentation des Textes

Suivant les étapes de prétraitement (extraction, sélection et transformation), un texte est représenté par un vecteur de caractéristiques. On parle alors de représentation du texte. La figure 2.9 présente la place de la représentation de texte dans le processus global de classification. Suivant le type des caractéristiques, une représentation de texte peut prendre plusieurs formes. La plus connue et la plus largement répandue est la représentation en sac de mots (Bag-Of-Words, BOW) (Harris, 1954) se basant sur le modèle vectoriel (Vector Space Model, VSM) (Salton and McGill, 1986). La représentation peut également être fondée sur une liste de N-Grams, de groupes nominaux ou encore de concepts. Enfin, elle peut être un mélange de plusieurs représentations telles que la représentation hybride compilant des concepts et des mots. Dans cette sous partie, je présente les principales méthodes de représentation, leurs avantages et leurs faiblesses.



FIGURE 2.9: États des textes lors d'une classification.

2.2.4.1 Représentation en Sac de Mots

La représentation en sac de mots est de loin la plus suivie dans la recherche et l'industrie. La plupart des articles de recherche sur la classification de textes se basent sur cette technique de représentation.

Comme son nom l'indique, cette représentation vise à représenter le texte avec une liste de mots simples contenus dans ce dernier. Suivant les différentes étapes d'extraction et de sélection, certains mots comme les mots vides ne font pas partis du sac. A chaque mot est associée une mesure. Cette mesure peut être une valeur binaire (présence = 1; absence = 0), une fréquence d'apparition dans le document ou encore le nombre d'occurrences dans le texte.

L'intérêt de cette représentation est sa simplicité et ses performances. De nombreux sous domaines de l'apprentissage tels que la catégorisation des objets 3D (Toldo et al., 2009), la détection de spams (Sasaki and Shinnou, 2005), le "topic modelling" (Hong and Davison, 2010) ou encore la biologie (Nobata et al., 1999) l'utilisent. Cette représentation permet la classification de textes écrits dans diverses langues telles que le Chinois (Lu et al., 2010), l'Arabe (Al-Harbi et al., 2008) ou encore le Français (Trinh, 2007).

Cependant cette représentation soulève plusieurs problèmes notamment identifiés par (Bloehdorn and Hotho, 2004) :

- La polysémie. Suivant le contexte, un mot peut avoir des sens différents. Par exemple, le mot "orange" peut se rapporter à la couleur, le fruit, la ville du sud de la France ou encore l'opérateur téléphonique. En suivant le paradigme du sac de mots, un article sur la ville d'Orange peut être catégorisé dans une catégorie traitant de fruits.
- La synonymie. Deux (ou plus) mots distincts peuvent avoir le même sens ou désigner la même chose. Par exemple, le mot "plectre" représente le même objet que le mot "médiator". Suivant la langue utilisée, un objet peut être traduit du latin et du grec et être ainsi décrit par plusieurs mots.

- Les liens sémantiques. Les mots proches sémantiquement sont considérés comme n'ayant aucun lien. Par exemple, les mots "animal" et "chat" sont liés par une relation hyperonymie mais restent considérés comme distincts l'un de l'autre.
- Division de groupes de mots. La représentation en sac de mots propose de décrire un texte par certains de ses mots. Si le texte contient la suite de mots "poste de police", les mots retournés seront "poste" et "police" en considérant que le mot vide "de" est supprimé automatiquement. Or le mot "poste" ne se réfère pas forcément au domaine de la police.

Par exemple, la représentation en sac de mots de la phrase "Kevin Parker est le leader de Tame Impala" est "Kevin", "Parker", "est", "le", "leader", "de", "tame", "impala".

2.2.4.2 Représentation en Sac de N-Grammes

La représentation en N-grammes n'est pas courante dans le domaine de la classification de textes. Cependant, certains comme (Cavnar and Trenkle, 1994) ou encore (Fürnkranz, 1998) ont proposé des solutions en utilisant ce paradigme. Un n-gramme est une séquence de n éléments. Ainsi, une séquence de 2 éléments est un 2-gramme aussi appelé bigramme. Dans le domaine du traitement du langage naturel, un n-gramme est une séquence de n mots, cependant les autres communautés s'accordent à le décrire comme une séquence de n caractères.

Cette représentation permet la représentation de textes écrits dans des langues ne distinguant pas les mots par des espaces comme le Chinois et le Japonais. Elle permet aussi d'identifier des similitudes entre les mots ayant la même racine. Par exemple, les mots "chanteur" et "chanter" sont distincts dans une analyse en sac de mots mais dans une analyse en 4-grammes par exemple, les deux mots seraient retournés sous la forme de la même séquence "chant".

Cependant, ce paradigme ne résout pas les différents problèmes de la représentation en sac de mots. Il ajoute même une limite : l'explosion de la dimension. Suivant la valeur de n, une représentation en n-grammes peut augmenter très fortement la dimension de l'analyse. Plus la valeur de n est petite, plus la dimension sera élevée. Or (Beyer et al., 1999) a montré qu'à partir d'un nombre suffisamment important de dimensions, les observations (des textes dans notre étude) deviennent équidistants et sont donc difficilement comparables.

Par exemple, pour $n = \{2, 3, 4\}$, les représentations en sac de n-grammes du texte "téléphone portable" sont :

- Avec $n = 2$: "té", "lé", "ph", "on", "e ", "po", "rt", "ab" et "le".
- Avec $n = 3$: "tél", "éph", "one", " po", "rta", "ble"
- Avec $n = 4$: "télé", "phon", "e po" et "rtab"

2.2.4.3 Représentation en Sac de Groupes de Mots

Un groupe de mots (phrase en anglais) est un n-gramme dont chaque élément est un mot. Un groupe de mots est donc un n-gramme du point de vue du traitement automatique du langage. Initiée par Lewis (Lewis, 1992b), la représentation en sac de groupes de mots reste peu utilisée pour la classification.

Deux méthodes permettent la création des groupes de mots. La première est statistique et vise à trouver les mots qui co-occurrent plus que d'autres dans les textes. Elle a notamment été utilisée par (Mladenic and Grobelnik, 1998) et (Koehn et al., 2003). La seconde se rapporte au traitement automatique des langues et vise à extraire les groupes de mots à partir de règles de syntaxes. Cette méthode a notamment été utilisée par (Lewis and Croft, 1989).

Cette méthode a l'avantage de considérer des relations entre les mots mais reste limitée si on ne considère que les groupes de mots excluant les mots seuls. De plus, les inconvénients du sac de mots ne sont pas traités par cette méthode.

2.2.4.4 Représentation en Sac de Concepts

Un concept est un élément représentant une connaissance. Il peut être vu comme étant l'élément de base d'une ontologie (Kehagias et al., 2003) ou comme la catégorie d'articles de Wikipédia[5] et autres bases de connaissance (Huang et al., 2009). Dans mes travaux, je considère le concept comme l'unité de base de l'ontologie. Un concept décrit un label et peut avoir des synonymes. Les labels et les synonymes sont des mots ou groupes de mots. Par exemple, le concept décrit par le label "acétaminophène" a pour synonyme "paracétamol". Deux concepts peuvent être liés par une ou plusieurs relations sémantiques. Par exemple, le concept "acétaminophène" est lié par une relation d'hyperonymie au concept "médicament".

Ce paradigme introduit une nouvelle donnée d'entrée : une ontologie. Plus d'information sur cet outil de représentation de la connaissance est donné dans la partie 2.1.

[5] http://wikipedia.com/

Représenter un texte avec une liste de concepts permet de passer outre les contraintes du sac de mots, à savoir : la polysémie, la synonymie, les mots complexes et les relations entre les caractéristiques. De plus, cette représentation permet de réduire fortement la dimension de l'analyse, l'ontologie regroupant les concepts pouvant être vue comme un filtre.

Malgré ses qualités face au paradigme du sac de mots, cette représentation n'améliore pas les résultats d'une classification comme l'a montré Kehagias (Kehagias et al., 2003). L'origine de ces mauvais résultats est la mauvaise couverture de l'ontologie sur les textes analysés. Une ontologie de domaine trop précise ne permet pas de capter les concepts plus généraux contenus dans les textes. A l'inverse, une ontologie de domaine approximative et globale ne permet pas l'extraction de concepts précis et discriminants.

2.2.4.5 Représentation Hybride

La représentation hybride vise à joindre les intérêts de la représentation en sac de concepts et les performances de la représentation en sac de mots. Un texte est ainsi représenté par les concepts qui le composent et par des mots, groupes de mots ou encore n-grammes contenus dans ce dernier.

En observant les mauvais résultats de l'approche en sac de concepts par (Kehagias et al., 2003), Blodehdorn dans son article (Bloehdorn and Hotho, 2004) a proposé une approche hybride. Ses conclusions confirment l'amélioration des résultats par les modèles hybrides.

2.2.4.6 Amélioration des Représentations

Les techniques de représentation en sac de concepts et hybrides ont apporté du sens à la représentation des textes. Cependant les résultats obtenus n'ont pas été encourageant et ces paradigmes restent minoritaires contre celui du sac de mots. Certains chercheurs ont cependant la conviction qu'une représentation avec des sens peut permettre à la machine de devenir un meilleur classifieur.

Ainsi, des recherches ont été menées pour tenter d'améliorer ces représentations. Trois techniques se distinguent :

- Le remplacement des mots et groupes de mots par des concepts contenus dans le texte. Cette méthode initiée par (Hotho et al., 2003) pour le clustering de textes permet l'enrichissement d'un sac de mots à l'aide d'une base de connaissances (en l'occurrence Wordnet (Miller, 1995)). Ainsi, si un mot extrait du texte peut être

identifié par un des concepts de la base de données, alors le mot est remplacé par celui-ci. Cette méthode évite l'accroissement de la dimension par des nouvelles caractéristiques.

- L'ajout de concepts contenus dans le texte. Cette technique a été présentée par (Benkhalifa et al., 2001) et vise à enrichir un sac de mots classique avec des connaissances de la base de connaissance WordNet. Dans son article, Benkhalifa se base sur un algorithme de classification de textes semi-supervisé. Les mots identifiés comme désignant un concept sont conservés. Contrairement à la méthode précédente, celle-ci accroît la dimension de l'analyse.
- L'ajout de concepts non contenus dans le texte. Cette méthode a pour objectif d'intégrer des connaissances non contenues initialement dans les textes à l'aide des relations sémantiques définissant l'ontologie. (Wang et al., 2007) a proposé d'enrichir un sac de mots avec les concepts des articles de Wikipédia.

2.2.5 Algorithmes d'Apprentissage Supervisé

2.2.5.1 Les Méthodes Transductives et Inductives

D'après (Tufféry, 2012), deux catégories de méthodes se distinguent de l'apprentissage supervisé : les méthodes transductives et inductives. Les premières permettent le classement d'observations non étiquetées à partir d'observations étiquetées sans étape préalable. Les secondes comprennent une étape de généralisation des données étiquetées pour ensuite, dans une seconde étape, classifier de nouvelles données non étiquetées.

L'apprentissage transductif reste minoritaire face à l'apprentissage inductif malgré sa mise en avant par Vapnik en 1998 (Vapnik and Vapnik, 1998). Les méthodes transductives sont généralement utilisées dans les applications d'apprentissage semi-supervisé où les données d'apprentissage ne sont pas toutes étiquetées. Dans ce cas de figure, une méthode transductive permet la labélisation des données non étiquetées à partir des données étiquetées du jeu d'apprentissage. Il existe quelques algorithmes d'apprentissage transductif. La plus connue de toutes est sans doute la méthode des k plus proches voisins (K-NN) décrites en 1967 dans (Cover and Hart, 1967). Certaines méthodes inductives ont été adaptées au modèle transductif, notamment la méthode de la machine à vecteurs de support (SVM) dans (Gammerman et al., 1998).

L'apprentissage inductif regroupe des méthodes d'apprentissage supervisé largement répandues dans la communauté. Contrairement aux méthodes transductives, les méthodes inductives comprennent une étape de généralisation des données d'apprentissage. On appelle cette généralisation un modèle d'apprentissage. C'est en s'appuyant sur ce dernier

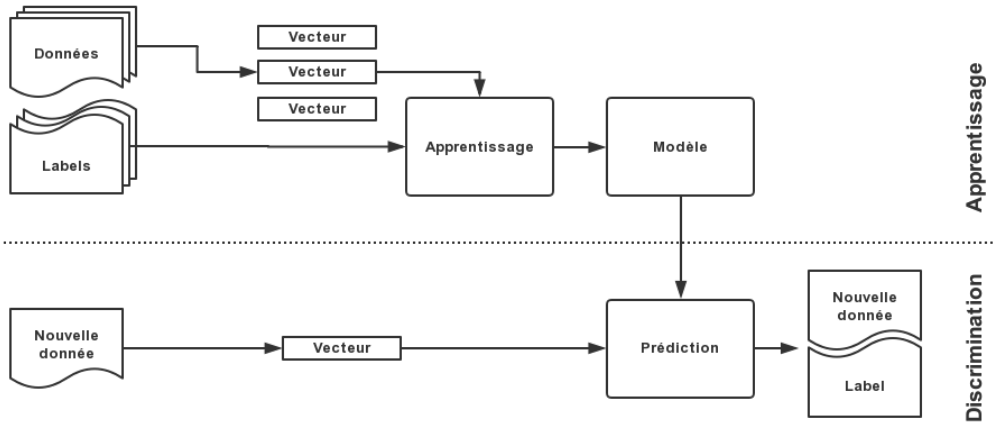


FIGURE 2.10: Principe général d'une méthode inductive.

que les nouvelles observations sont discriminées suivants les différents labels proposés dans le corpus d'apprentissage. 2.10 présente le fonctionnement général d'une méthode inductive. L'apprentissage inductif regroupe de nombreuses méthodes de différents types : les méthodes de réseau neuronal artificiel tels que le perceptron et le réseau à fonction radiale de base, les méthodes à base de règles logiques tel que l'arbre de décision, les méthodes à base de fonctions mathématiques tel que le SVM, les méthodes probabilistes tel que le classifieur naïf Bayésien et les méthodes d'agrégation de modèles (bagging) tel que les forêts aléatoires (Breiman, 2001).

Les parties suivantes présentent les principaux modèles transductifs utilisés pour la classification de textes.

2.2.5.2 Machine à vecteurs de support

Une machine à vecteurs de support également nommée *séparateurs à vaste marge* et simplifié par l'acronyme anglais SVM est une méthode d'apprentissage supervisée pour la classification et la régression généralisant la méthode de régression linéaire. Elle a été introduite par Vapnik en 1995 dans (Vapnik, 2013) et (Cortes and Vapnik, 1995). Ce dernier s'est inspiré des travaux sur les hyperplans à marge maximale (Vapnik, 1963) et (Duda et al., 2012) et sur ceux des fonctions noyaux (Mercer, 1909). Dès 1998, l'algorithme a été adapté à la classification de textes (Gammerman et al., 1998). (Wang, 2005) propose une revue complète de la méthode et de ses applications.

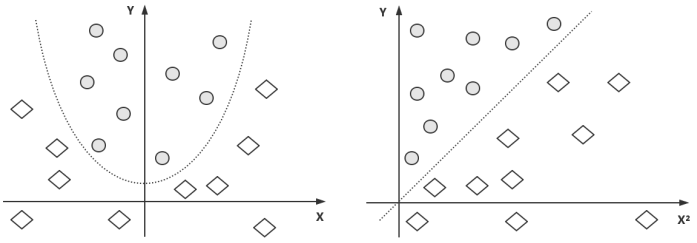


FIGURE 2.11: Changement d'espace pour obtenir une séparation linéaire.

La méthode du SVM a été initialement créée pour répondre à des problèmes de classification binaire. Cette dernière reste malgré tout adaptable à la classification multi-classes. En effet, différentes méthodes permettent le passage d'un problème multi-classes à binaire. Les deux techniques les plus souvent utilisées sont one-versus-all (Rifkin and Klautau, 2004) et one-versus-one (Duan and Keerthi, 2005). (Hsu and Lin, 2002) propose une étude comparative des méthodes multi-classes pour le SVM. Ainsi, la présentation de l'algorithme peut se résumer au cas de la classification binaire.

L'objectif du SVM est la recherche d'un hyperplan de marge maximale séparant les données d'apprentissage des deux classes de l'analyse. La recherche de cet hyperplan est fondée sur l'ensemble des fonctions linaires séparant distinctement les classes A et B du type $f(X) = A.X + b$ avec X le vecteur d'entrée à n dimensions $X = (x_1, x_2, ..., x_n)$, A le vecteur de poids $A = (a_1, a_2, ..., a_n)$ et b le biais. Si aucune fonction linéaire ne peut séparer les deux classes, il est nécessaire de procéder à une transformation de l'espace de l'analyse. L'idée est d'ajouter des dimensions jusqu'à ce que les classes soient séparables. Cette opération est réalisée à partir d'une fonction noyau. La figure 2.11 présente une transformation de l'espace permettant de séparer deux ensembles de vecteurs qui ne l'étaient pas dans le plan initial. Cette capacité de changement d'espace est l'une des deux briques fondatrices de l'algorithme.

Pour la suite des explications de l'algorithme, nous nous plaçons dans le cas où les classes sont séparables linéairement. Considérons que $f(X)$ est une fonction séparatrice. Si $f(X) > 0$ alors le vecteur X est classé dans la catégorie A, si $f(X) <= 0$ alors X est classé dans la catégorie B. Les points les plus proches de $f(X)$ pour chacune des deux classes sont appelés les vecteurs de support. La fonction $f(X)$ est optimale si elle maximise la marge entre elle-même et les vecteurs de support. Sur la figure 2.12, h et g sont deux fonctions linéaires séparants les groupes A et B. L'hyperplan $h(X) = 0$ est optimal car sa marge associée est maximale. L'hyperplan $g(X) = 0$ est quant à lui correct mais non optimal.

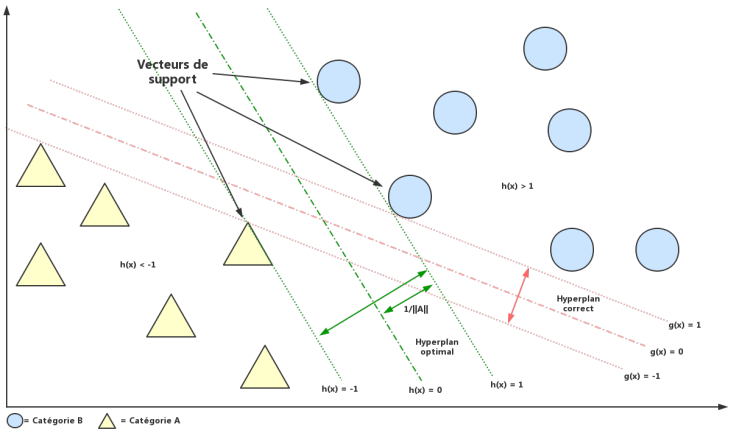


FIGURE 2.12: Marge optimale (la plus large) en vert et marge correcte mais non optimale en rouge.

2.2.5.3 Réseau de Neurones

Le réseau de neurones artificiels (abrégé réseau de neurones) est l'un des algorithmes les plus utilisés de l'apprentissage automatique. Cet algorithme s'adapte à l'apprentissage non supervisé, supervisé pour la régression comme la classification (Specht, 1991). Il est généralement non probabiliste mais Specht a proposé une version qui l'est (Specht, 1990). Le réseau de neurones est l'algorithme d'apprentissage à la base du deep learning (LeCun et al., 2015).

Le réseau de neurones artificiels est une tentative de formalisation des réseaux de neurones naturels et notamment du neurone formel. Sa création n'est pas due à l'informatique uniquement mais aussi aux sciences du vivant et de l'Homme. (Lettvin et al., 1959) est l'article fondateur du réseau de neurones.

Le réseau de neurones peut être vu comme un graphe contenant des nœuds répartis par couche. Un réseau de neurones est constitué de trois couches au moins. La première couche est la couche d'entrée. Elle contient autant de nœuds que de variables définissant le vecteur de l'individu statistique. La seconde est la couche dite cachée. Il y a au moins une couche cachée mais il peut y en avoir de nombreuses (c'est notamment le cas dans les algorithmes de deep learning). Enfin, la dernière couche correspond à la couche de sortie. Dans le cas d'une régression, il n'y a qu'un nœud de sortie, dans le cas d'une classification, il y a autant de nœuds de sortie que de classes. Le schéma 2.13 présente la topologie d'un réseau de neurones à trois entrées et deux sorties et contenant une unique couche cachée de quatre nœuds. Il n'existe pas de formule permettant de calculer le nombre optimal de couches cachées et de neurones dans chacune d'elles. Cependant (Tufféry, 2012) rappelle que le nombre de nœuds des couches cachées est généralement

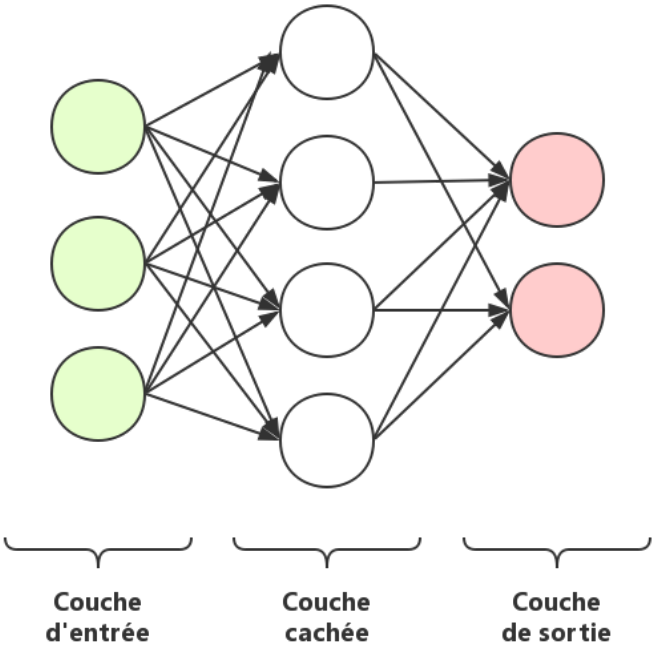


FIGURE 2.13: Réseau de neurones simplifié à deux sorties (classification).

borné par $n/2 < m < 2n$ avec n le nombre de nœuds en entrée et m le nombre de nœuds des couches cachées. Aussi, Tufféry indique que le nombre de nœuds de la couche cachée dépend également du nombre d'observations en entrée tel que $N > 5m(n+k)$ avec N le nombre d'individus de l'analyse et k le nombre de classes.

Suivant sa couche d'appartenance, un nœud a un rôle particulier. Les nœuds de la couche d'entrée n'ont aucune réelle fonction mis à part le transfert de l'information vers les nœuds de la couche cachée. Les nœuds de la couche cachée et de la couche de sortie ont deux fonctions distinctes. La première est de produire un résumé des informations en entrée en réalisant la somme des $n_i p_i$ où n_i est la valeur retour du nœud i et p_i son poids accordé. La seconde fonction d'un nœud est de calculer une valeur retour. Pour cela, un nœud utilise une fonction de transfert (généralement la fonction sigmoïde logistique). En minimisant l'erreur (taux de mauvaises classifications), l'algorithme ajuste les différents p_i, on dit qu'il ajuste les poids. C'est ici tout le rôle de l'apprentissage.

La fonctionnement d'un neurone (hors couche d'entrée) est explicité par le schéma 2.14.

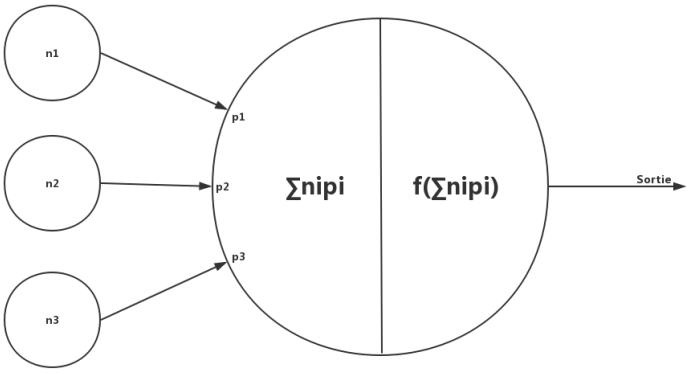


FIGURE 2.14: Principe de fonctionnement divisé en deux parties d'un neurone.

2.2.5.4 Arbre de décision

Un arbre de décision est une méthode d'aide à la décision historiquement réalisée manuellement. Il existe des algorithmes permettant de les construire automatiquement. (Safavian and Landgrebe, 1990) présente un état de l'art complet sur les arbres de décision pour la classification automatique. Ces derniers sont des algorithmes très appréciés de la communauté grâce notamment à sa représentation simple du modèle de classification sous forme d'arbre facile à comprendre et expliquer (même pour les non experts). Pourtant, la méthode propose souvent des performances mitigées par rapport à d'autres algorithmes.

L'arbre de décision est un ensemble de nœuds et d'arcs orientés. Les nœuds internes sont appelés "nœuds de décision" et sont associés à une ou plusieurs règles. Une règle concerne une ou plusieurs variables décrivant les observations. Les arcs issus de ces nœuds sont les solutions de ces règles. Les feuilles de l'arbre (nœuds aux extrémités de l'arbre) correspondent aux différentes classes de l'analyse.

La phase d'apprentissage du modèle consiste en l'apprentissage des règles les plus discriminantes. Généralement, un nœud est associé à une seule règle mais certaines méthodes proposent d'y associer plusieurs. Les règles peuvent concerner des variables catégorielles ou numériques, ce qui fait de la méthode de l'arbre de décision, une méthode très générique.

La phase de classification est un parcours simple de l'arbre. Le nœud de départ est le nœud racine et correspond à la première règle (la plus discriminante). En fonction des données de l'observation à labéliser, le parcours du graphe aboutit sur une feuille et la catégorie associée à celle-ci est désignée comme celle décrivant l'observation.

ID	Âge	Habitat	Revenu mensuel	Voiture
1	27	citadin	1500	Citadine
2	42	contadin	2300	4x4
3	18	contadin	1800	4x4
4	19	citadin	1100	Citadine
5	29	citadin	7400	Berline
6	36	contadin	4500	4x4
7	39	citadin	8200	Berline
8	61	citadin	6200	4x4
9	59	citadin	5400	4x4
10	26	citadin	7400	Berline

FIGURE 2.15: Données d'apprentissage pour l'arbre de décision.

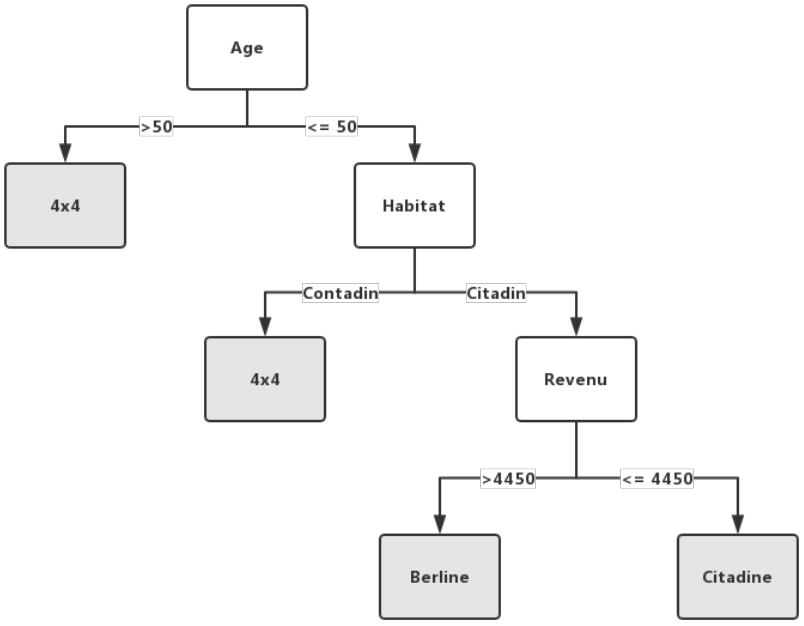


FIGURE 2.16: Arbre de décision possible construit à partir des données d'apprentissage.

Soit un ensemble de 10 observations décrites par l'ensemble V de 4 variables. Soit V = {"Âge", "Habitat", "Revenu mensuel", "Voiture"}. On cherche à prédire la variable *Voiture*. Le tableau 2.15 présente notre ensemble de données d'apprentissage.

La variable à prédire étant *Voiture*, les feuilles de l'arbre désignent donc les différentes valeurs que prend celle-ci. La figure 2.16 est un des modèles possibles représentant ce corpus d'apprentissage.

A partir de ce modèle, il est désormais possible de prédire par quelle voiture sera intéressée une nouvelle personne en fonction de son âge, son habitat et son revenu. Par exemple,

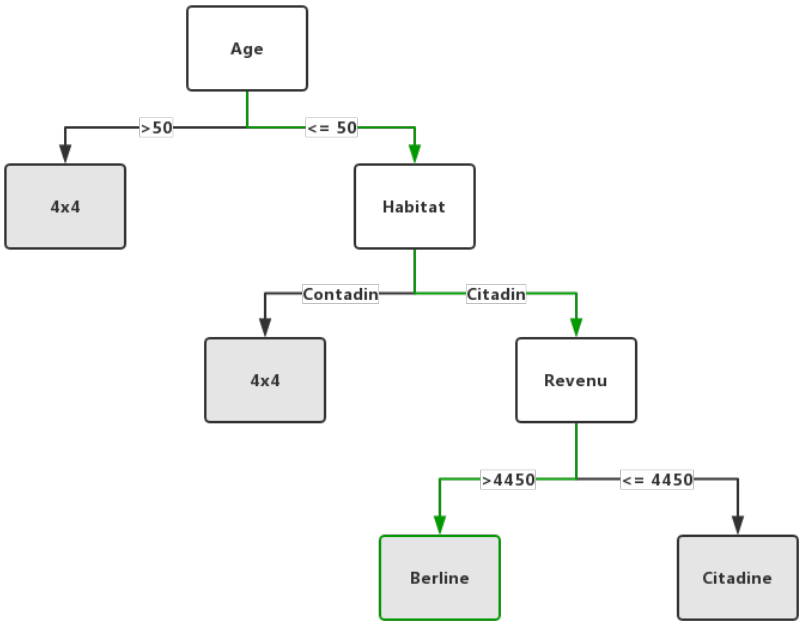


FIGURE 2.17: Prédiction d'un nouvel individu.

un concessionnaire automobile ayant accès à cet arbre de décision, proposera en priorité une berline à son client de 45 ans, habitant Paris et ayant un revenu de 6200*euros*. La figure 2.17 présente le chemin parcouru par l'algorithme pour arriver à cette prédiction.

2.2.5.5 Forêt d'Arbres Décisionnels

La forêt d'arbres décisionnels (couramment appelé random forest) est un algorithme de classification et de régression développé par Leo Breiman dans son article (Breiman, 2001). Son algorithme est inspiré des travaux de (Ho, 1995) et (Dietterich, 2000). L'objectif fondamental de la forêt d'arbres décisionnels est de produire une famille d'arbres de décision à partir de différents sous-ensembles de données et de variables. Cet algorithme reprend les principes du bagging (Breiman, 1996) et du boostrap (Metropolis and Ulam, 1949) et les applique à la méthode des arbres de décisions.

Le bagging consiste en la création d'une famille de modèles qu'on estime pour chacun moyennement performant et instable (un réseau de neurones, un arbre de décision, etc.), augmentant considérablement la robustesse des modèles. La prédiction d'une méthode de bagging correspond à la prédiction "moyenne" de l'ensemble de cette famille de modèles (plus précisément, il s'agit d'un vote dans le cas de la classification et d'une moyenne dans le cas d'une régression).

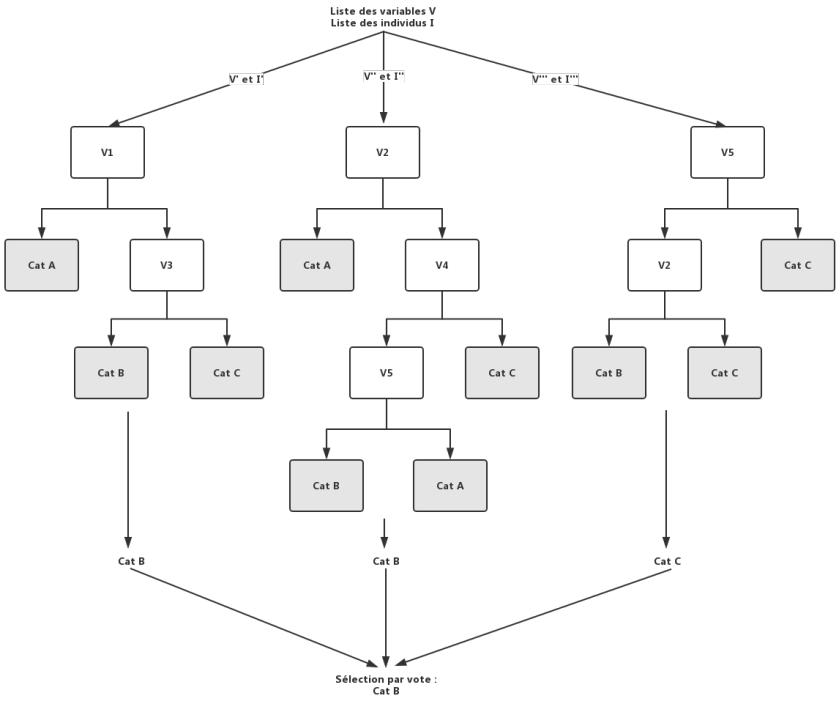


FIGURE 2.18: Forêts de trois arbres décisionnels avec sélection des variables et des individus.

Le boostrap est une méthode de génération de sous-échantillons aléatoires à partir d'un échantillon de base et d'agrégation des résultats. Chaque sous échantillon est créé à partir de l'échantillon initial avec remise (une observation peut se retrouver une ou plusieurs fois dans le même sous échantillon ou n'être présente dans aucun des sous-échantillons). Cette méthode permet d'évaluer la variabilité du modèle.

La figure 2.18 présente un exemple de la méthode avec trois arbres de décision appliqué au problème de classification. Soient I l'ensemble des individus et V l'ensemble des variables explicatives. Le premier arbre a été construit à partir du sous-ensemble I' de I et V' de V le second à partir du sous-ensemble I'' de I et V'' de V et le dernier à partir du sous-ensemble I''' de I et V''' de V. La variable explicative est $P = \{"CatA", "CatB", "CatC"\}$.

Un inconvénient de la méthode Random Forest face à celle de l'arbre de décision est la perte en lisibilité des modèles et donc en interprétation de ces derniers.

Pour plus de détails sur la classification et la régression à l'aide de la méthode de random forest, le lecteur est invité à se référer à (Liaw and Wiener, 2002).

2.2.5.6 Classifieur Naïf Bayésien

Le classifieur naïf Bayésien (Lewis, 1998) est un algorithme appartenant à la catégorie des méthodes de classification Bayésienne probabiliste fondée sur le théorème de Bayes. La méthode suppose que les caractéristiques descriptives sont indépendantes, ce qui est généralement faux. C'est en se basant sur cette hypothèse que cette dernière est dite "naïve". Deux modèles de classifieur naïf Bayésien se distinguent : le modèle multivarié de Bernoulli et le modèle multinomial. La principale différence entre ces deux derniers est que le modèle multivarié de Bernoulli ne prend pas en compte la fréquence des termes au sein des documents contrairement au modèle multinomial. Ces deux méthodes ayant souvent été source de confusion entre elles, (McCallum and Nigam, 1998) a publié une comparaison des modèles. Par soucis de simplicité, nous avons choisi d'utiliser l'algorithme multivarié de Bernoulli pour notre expérience.

L'objectif de cet algorithme est le calcul des probabilités à posteriori qu'un document appartienne à chacune des catégories de l'analyse. La classe ayant la plus forte probabilité est sélectionnée comme label du document. Soient k le nombre de classes de l'analyse et $V = \{t_1, ..., t_n\}$ l'ensemble des n termes extraits de l'apprentissage. On note $Q = \{t_{i1}, ..., t_{im}\}$ un sous-ensemble de termes du sac de mots. On appelle C^T la classe associée au sous-ensemble T de V. La probabilité à posteriori que la classe i soit associée au document dont le sac de mots est Q est $P(C^T = i|T = Q)$. La catégorie sélectionnée pour labélliser le document est obtenue par $\arg\max_i P(C^T = i|T = Q)$. Pour la suite de l'explication, nous simplifions $P(C^T = i|T = Q)$ par $P(C|Q)$.

Le théorème de Bayes nous donne l'équation suivante :

$$P(C|Q) = \frac{P(C)P(Q|C)}{P(Q)} \tag{2.8}$$

Cette formule appliquée à la classification de textes se simplifie en supprimant le dénominateur, ce dernier devenant une constante, d'où $P(C|Q) = P(C)P(Q|C)$. En faisant l'hypothèse de l'indépendance des m termes de Q, on obtient la formule de classification suivante :

$$P(C|Q) = P(C)\prod_{j=1}^{m} P(t_j|C) \tag{2.9}$$

Pour plus de détails sur la méthode, le lecteur peut se référer à (Aggarwal and Zhai, 2012).

Concernant l'apprentissage du classifieur, le principe est d'estimer les $\widehat{P}(t_j|C_i)$ des probabilités $P(t_j|C_i)$. La méthode d'apprentissage est fondée sur l'estimation du maximum de vraisemblance. L'estimation du terme t_j associée à la classe C_i est calculée par

$$\widehat{P}(t_j|C_i) = \frac{count(t_j, C_i) + 1}{\sum_{t \in V}(count(t, C_i)) + |V|} \tag{2.10}$$

Pour éviter que le dénominateur soit égal à 0, la méthode classique est l'utilisation de la technique dite de LaPlace Smoothing. L'estimation de probabilité devient :

$$\widehat{P}(t_j|C_i) = \frac{count(t_j, C_i) + 1}{\sum_{t \in V}(count(t, C_i) + 1)} \tag{2.11}$$

$$\widehat{P}(t_j|C_i) = \frac{count(t_j, C_i) + 1}{\sum_{t \in V}(count(t, C_i)) + |V|} \tag{2.12}$$

2.2.6 Évaluation du Modèle de Classification

Il n'existe pas d'algorithme parfait. Ceux présentés ci-dessus sont ceux qui présentent les meilleures performances de manière générale lorsqu'ils sont appliqués à la classification supervisée de textes. De la même façon, certains de ces algorithmes sont paramétrables et la modification d'un des paramètres peut changer le comportement global du modèle de classification résultant. Sélectionner le meilleur modèle parmi l'ensemble disponible nécessite d'être en capacité de les évaluer.

Évaluer un modèle signifie comparer les catégories prédites avec les réelles. Un ensemble de textes pré labélisés est donc indispensable. Il est d'usage de diviser le corpus de textes d'apprentissage en deux sous-ensembles. Le premier permet l'apprentissage et le second le test du modèle. Une autre découpe divise le corpus initial en trois sous-ensembles, ajoutant une division de validation. Ajouter ce sous-ensemble permet de ne pas créer un modèle précisément adapté au sous-ensemble de test (il s'agit en quelque sorte d'éviter le sur apprentissage sur l'ensemble de test). Ce principe général de division du corpus de texte est utilisé par les deux grandes méthodes de validation : la validation "holdout" et la validation croisée. La sortie de chacune de ces deux validations est une matrice de confusion présentant en colonne le nombre de classifications dans les classes prédites et en colonne le nombre de classifications dans les classes réelles. Cette dernière permet de calculer des indices de performances. Le schéma 2.19 présente le fonctionnement générale de l'étape de validation d'une classification supervisée.

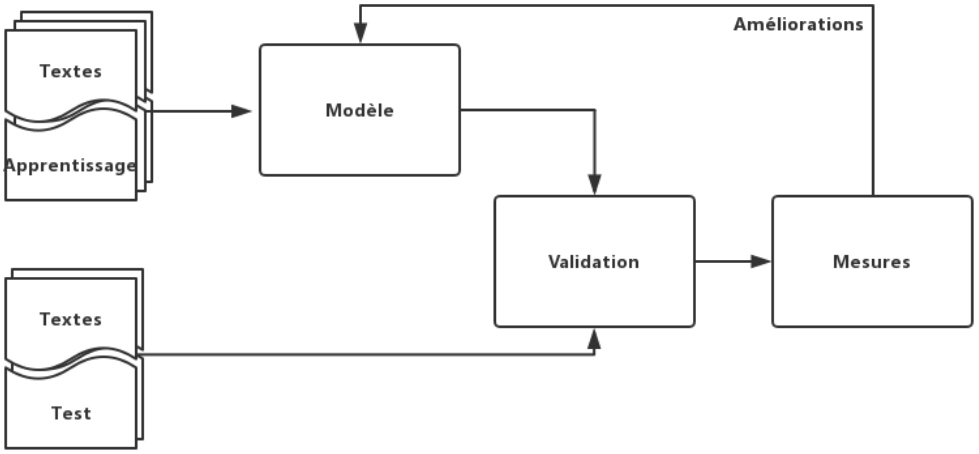


FIGURE 2.19: Validation d'un modèle de classification supervisée.

Le deux sous-sections suivantes détaillent les méthodes de validation "holdout" puis de validation croisée. Une troisième sous section présente les principaux indices de performance permettant d'évaluer un modèle de classification.

2.2.6.1 Validation holdout

La méthode "holdout" est la plus simple à mettre en place. L'ensemble de textes est divisé en deux sous-ensembles. Le premier contient environ 70% des textes et est réservé à l'apprentissage du modèle. Le second contient les textes restant (environ 30%) et sert au test de ce dernier. La répartition des deux sous-ensembles se fait de manière aléatoire afin d'éviter tout ordre au sein du corpus entachant la généralisation.

2.2.6.2 Validation Croisée

La méthode de validation croisée est plus fine que "holdout". Celle-ci propose de diviser l'ensemble de textes pré-labellisés en k sous-ensembles de taille équivalente. L'un des sous-ensembles est sélectionné comme étant celui de test et les $(k-1)$ restants sont réservés à l'apprentissage. L'opération est exécutée k fois de manière à ce que chacun des sous-ensembles ait été désigné comme ensemble de test. Cette méthode permet de généraliser le modèle en évitant tout sur-apprentissage sur le corpus d'apprentissage et celui de test.

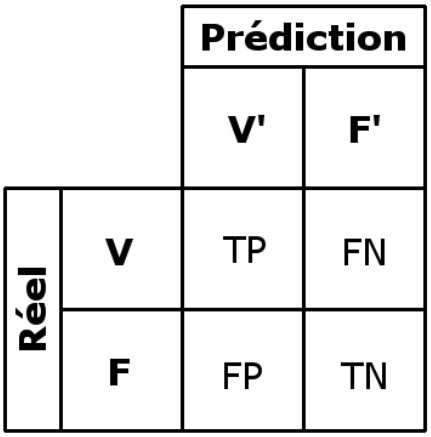


FIGURE 2.20: Informations obtenues par une table de confusion.

2.2.6.3 Mesure d'Évaluation

Que l'on utilise la validation "Holdout" ou croisée, la sortie de ces dernières est une matrice de confusion. Une matrice de confusion est une matrice carré de dimension k (nombre de catégories de l'analyse) présentant en ligne les occurrences classes réelles des textes et en colonne les occurrences des classes prédites. Cette matrice se résume par une table de confusion. Une table de confusion peut être vue comme une matrice de confusion d'une analyse à deux classes telle qu'une classification de spams (une classe "spam" et une classe "non spam"). La figure 2.20 présente le schéma général de cette dernière. Quatre informations sont extraites de la matrice de confusion :

- TP (True Positive ou vrai positif) : Un texte est un vrai positif si la catégorie prédite est V et que sa vraie catégorie est V.
- TN (True Negative ou vrai négatif) : Un texte est un vrai négatif si la catégorie prédite est F et que sa vraie catégorie est F.
- FP (False Positive ou faux positif) : Un texte est un faux positif si la catégorie prédite est V alors que sa vraie catégorie est F.
- FN (False Negative ou faux négatif) : Un texte est un faux négatif si la catégorie prédite est F alors que sa vraie catégorie est V.

A partir de cette table de confusion, plusieurs indices peuvent être calculés afin de mesurer la qualité du modèle. Comme présenté dans (Yang, 1999) et initié par (Rijsbergen, 1975),

les quatre principales mesures sont le taux d'erreur, la précision, le rappel (aussi appelé la sensibilité) et le F1 score.

Le taux d'erreur noté $ErrorRate$ est le pourcentage de mauvaises prédictions sur l'ensemble des prédictions. Il est défini par :

$$ErrorRate = \frac{FP + FN}{TP + TN + FP + FN} \tag{2.13}$$

La précision correspond au pourcentage de bonnes prédictions V' sur la somme des prédictions V' bonnes ou mauvaises. On la calcule par :

$$Precision = \frac{TP}{TP + FP} \tag{2.14}$$

Le rappel correspond au pourcentage de bonnes prédictions V' sur l'ensemble des textes réellement attribués à la catégorie V. On le note :

$$Rappel = \frac{TP}{TP + FN} \tag{2.15}$$

Le F1 score est une mesure dérivée de la précision et du rappel. Elle est donnée par :

$$F1score = \frac{2 * Precision * Rappel}{Precision + Rappel} \tag{2.16}$$

Il existe d'autres indices pour mesurer la qualité d'un modèle, cependant ces quatre mesures suffisent à évaluer correctement un modèle.

2.2.7 Corpus de Textes

Afin de tester l'ensemble de ces algorithmes et techniques, il est nécessaire de disposer de matières premières : des textes labellisés. Obtenir un corpus de textes labellisés peut parfois être une étape compliquée car ces derniers sont coûteux à produire. En effet, chaque texte doit être labellisé manuellement pour permettre à la machine de retrouver le modèle. De plus, afin de comparer les résultats des différentes méthodes et techniques, il est nécessaire de disposer de corpus libres. C'est pour ces raisons que la communauté a mis en place plusieurs corpus dont les principaux sont présentés ci-après :

- Reuters-21578. Cet ensemble de données est de loin le plus utilisé dans la littérature. Il est la version améliorée des ensembles Reuters-22173 et Reuters-21450

(qui ont eux-mêmes été utilisés dans des recherches sur la classification de textes telles que (Apté et al., 1994), (Cohen and Singer, 1999), (Ng et al., 1997) ou encore (Wiener et al., 1995)). Les textes contenus sont des articles extraits du fil d'actualité Reuters de l'année 1987.

Dans ce corpus de 21578 textes, seulement 12902 d'entre eux sont assignés dans au moins une des 135 catégories. Pour plusieurs raisons, ce découpage n'a pas satisfait l'ensemble de la communauté et plusieurs chercheurs ont proposé leur propre découpage. Ainsi, (Joachims, 1998), (Baker and McCallum, 1998) et encore (Lam and Lai, 2001) ont utilisé une version de Reuters-21578 ayant une répartition en 90 catégories, (Bennett, 2003), (Tong and Koller, 2001) et aussi (Nigam et al., 2000) ont utilisé une version avec seulement 10 catégories et enfin, (Benkhalifa et al., 2001), (Dumais et al., 1998) et (Nardiello et al., 2003) ont utilisé une version avec 135 classes. Ce découpage ne convenant pas à tous, de nombreux chercheurs ont créé leur propre division. Les trois principales sont revues et comparées dans (Debole and Sebastiani, 2005).

Cet ensemble de données a plusieurs inconvénients. Premièrement, les textes ne sont pas répartis équitablement. Certaines catégories contiennent plusieurs milliers de textes alors que d'autres en contiennent seulement quelques-uns. Aussi, certaines classes sont sémantiquement liées entre elles (notamment les catégories "WHEAT" et "GRAIN" qui sont liées par une relation hiérarchique).

Ce corpus a notamment été utilisé par (McCallum, 1999), (Rennie et al., 2003), (Scott and Matwin, 1998) ou encore (Debole and Sebastiani, 2004).

- 20 Newsgroups (Lang, 1995). Ce corpus contient près de 20000 textes extraits d'articles de Usenet. Chaque texte est labellisé par une des 20 catégories.

  L'inconvénient de ce corpus est le sujet des différentes classes. Certaines sont très proches sémantiquement (par exemple "comp.sys.mac.hardware" et "comp.sys.ibm.pc.hardware") alors que d'autres n'ont aucun lien entre elles (par exemple "rec.sport.baseball" et "soc.religion.christian").

  Cet ensemble de textes a notamment été utilisé par (Kamvar et al., 2003), (Baker and McCallum, 1998) et (Kim et al., 2006).

- Ohsumed (Hersh et al., 1994). Cet ensemble de textes datant de 1991 compile des abstracts médicaux récoltés par l'université de science et de la santé d'Oregon. Ce dernier est disponible en deux versions. La première contient 50000 textes contre 20000 pour la seconde. Chaque texte est labellisé par une ou plusieurs des 23 catégories.

  Ce corpus est dépendant du domaine médical, en effet chaque catégorie correspond à une des 23 rubriques médicales de MeSH (Lipscomb, 2000). Cette particularité

peut être un atout dans le cas de classification de domaine ou un inconvénient dans celui de la classification générale.

Au travers des années, cet ensemble a largement été utilisé par la communauté. Citons par exemple (Toumouh et al., 2006), (Joachims, 1998) et (Dobrynin et al., 2005).

- Google Snippets (Phan et al., 2008). Ce corpus produit par Google[6] regroupe 12000 petits textes d'environ 13 mots chacun. Les textes sont répartis parmi 8 catégories ("Business", "Computers", "Culture-Arts-Entertainment", "Education-Science", "Engineering", "Health", "Politics-Society" et "Sports"). Ce corpus est plus récent que les autres et reste peu utilisé dans la communauté. En effet, les recherches sur les textes courts sont généralement menées sur des ensembles de tweets.

- Movie Reviews (Pang et al., 2002). Cet ensemble de textes contient 1400 revues de films distribués dans 2 catégories (700 revues positives et 700 revues négatives). Deux ans après sa publication, Pang a proposé une version enrichie contenant un total de 2000 revues (Pang and Lee, 2004).

  Cet ensemble de données est essentiellement destiné à l'analyse de sentiments et a été utilisé par entre autres (Kennedy and Inkpen, 2006), (Chaovalit and Zhou, 2005) et (Lin and He, 2009).

2.3 Mégadonnées

2.3.1 Définition en 5V

Le 22 août 2014, le Journal Officiel de la République Française publiait[7] la définition officielle de mégadonnées : "Données structurées ou non dont le très grand volume requiert des outils d'analyse adaptés.". Cette définition ayant pour objectif de franciser l'expression "Big Data" est intéressante. Elle apporte deux éléments de définition (la structure et le volume des données) et intègre la notion de solution (outils adaptés). Cependant cette définition est malheureusement imprécise car elle oublie le concept de vitesse d'exécution des traitements. En effet, les outils actuels sont capables de calculer les données aussi volumineuses soient-elle, le problème de ces outils étant le temps d'exécution nécessaire.

[6] http://google.com/

[7] Texte numéro 89 du journal numéro 0193 visible à l'adresse suivante : https://www.legifrance.gouv.fr/affichTexte.do;jsessionid=70E3FF9A4475F00957ACC60D0203F562.tpdila19v_2?cidTexte=JORFTEXT000029388087&dateTexte=&oldAction=rechJO&categorieLien=id&idJO=JORFCONT000029387119

Pourtant, Doug Laney publiait en 2001 dans un rapport de recherche de Gartner[8] (nommé à l'époque META Group) les nouvelles problématiques de l'analyse de données. Celles-ci s'exprimaient en 3 mots : Volume, velocity (vitesse) et variety (variété). L'expression des 3V était née et n'allait pas tarder à devenir très populaire. Ci-après, je décris ces 3Vs.

2.3.1.1 Volume

Les ordinateurs, les smartphones, les capteurs et maintenant les objets connectés produisent un nombre toujours plus grand de données. Chaque seconde, c'est 29000 Go de données qui sont publiées dans le monde[9]. L'avancée technologie du domaine informatique de ces dernières décennies (et notamment la popularité grandissante du cloud qui permet la délocalisation complète et transparente du matériel d'archivage) permet le stockage de ces nombreuses données. Ainsi, une entreprise de taille raisonnable capitalise des exaoctets de données dans ses bases. Le 'V' de volume fait référence à ces montagnes de données qui ne cessent de grandir de plus en plus rapidement à mesure que la technologie s'installe dans nos vies.

2.3.1.2 Variété

L'analyse de données classique se focalise sur des ensembles de données structurées. Il est d'usage de traiter des données stockées sous formes de fichier XML, CSV ou encore archivées dans des bases de données relationnelles. Ces analyses permettent de répondre à des problématiques figées et souvent peu complexes. La culture Open Source facilite l'accès à de nombreuses données. Citons les données de santé, météorologiques ou encore économiques[10]. Cependant, ces ensembles de données ont chacun leur structure. La variété fait référence à l'intégration dans un même système de ces ensembles différemment structurés.

2.3.1.3 Vitesse

Les capteurs sont de plus en plus performants et sont capables de produire des données avec une fréquence de plus en plus élevée (la vitesse de production est d'ailleurs l'un des facteurs de la voluminosité des données). Du coté applicatif, il y a une demande croissante

[8] Disponible à l'adresse suivante : http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

[9] Source : http://www.planetoscope.com/Internet-/1523-informations-publiees-dans-le-monde-sur-le-net-en-gigaoctets-.html

[10] Les ensembles de données cités sont disponibles depuis le site web https://www.data.gouv.fr/fr/.

d'analyse en temps réel. L'exemple le plus connu est sans doute le suivi d'échanges boursiers. Sans aller jusqu'à l'analyse en temps réel, les données sont devenues tellement volumineuses que les machines (même puissantes) mettent plusieurs jours à produire le résultat attendu, ralentissant lourdement le travail des analystes. Le concept de vitesse est le concept central de ce qu'on considère comme le domaine du Big Data. Les deux autres concepts pouvant être résolus de manière classique si l'analyste détient un temps infini pour résoudre son problème.

2.3.1.4 Véracité et Valeur

Quelques années plus tard, certains acteurs du Big Data ont proposé 2 nouveaux concepts commençant eux aussi par la lettre 'V' : Veracity (véracité) et Value (valeur). Cette définition en 5V n'a remporté qu'un succès modéré, souvent assimilé à un effet marketing plus qu'à une réalité terrain. La véracité des données est un concept fondamental pris pour acquis dans les analyses classiques. En effet, lorsque l'on souhaite analyser le contenu de l'encyclopédie en ligne Wikipedia[11] par exemple, il est implicitement dit que ce qui est contenu dans les pages de Wikipédia est valide. La réalité n'est pas si parfaite et de nombreuses informations (tweets, news, etc) sont analysées par des machines ne sachant pas détecter le vrai du faux. Cette discussion de véracité des données a également été introduite par les problématiques de détection de l'ironie. L'ironie est souvent perçue facilement par un humain mais reste difficilement détectable par une machine. Certains sites de fausses informations se sont fait une spécialité des articles ironiques. Citons pour les plus connus écrits en Français : Le Gorafi[12], La Dèche du Midi[13], etc. Dans l'optique de détecter l'ironie et ainsi augmenter la véracité des modèles d'apprentissage, certains chercheurs ont proposé des pistes de solution (Carvalho et al., 2009).

Le dernier 'V' fait référence à la valeur qu'un projet Big Data peut apporter. L'objectif est de tirer profit d'une analyse Big Data. Ce concept est beaucoup plus haut niveau que les précédents.

Pour répondre à ces contraintes, une solution a émergé suite à la publication par Google des articles de recherche MapReduce (Dean and Ghemawat, 2008) et Big Table (Chang et al., 2008). Ces papiers introduisent la distribution des données au sein d'un cluster informatique. Un cluster étant un ensemble de machines mises en réseau. Dans les sous parties suivantes, je présente les différentes solutions en termes de stockage distribué de données ainsi qu'en termes de traitements distribués des données.

[11] http://www.wikipedia.fr/index.php
[12] http://www.legorafi.fr/
[13] https://ladechedumidi.com/

2.3.2 Le Stockage de Données

Les premières bases de données informatiques sont apparues dans les années 1960. (Berg et al., 2013) propose un résumé de l'histoire des bases de données. Dans cette sous partie, j'introduis la notion de base de données puis je fais également un historique de cette dernière. Ensuite, je présente le célèbre modèle relationnel dans le détail. J'introduis les limites de ce dernier qui ont amené à la démocratisation des bases de données non relationnelles (dites NoSQL) que je décris dans un dernier paragraphe.

2.3.2.1 Définition

Une base de données est un outil permettant le stockage des données de manière plus ou moins structurée. Chaque base de données est organisée par un système de gestion de base de données (SGBD), son logiciel central. Ce dernier a pour objectif de faciliter la manipulation des enregistrements (consultation, filtrage, ajout, suppression, mise à jour, copie, etc.) par l'utilisateur. Le SGBD a également un rôle sécuritaire notamment en faisant respecter un fichier des différents droits (telle personne a le droit de faire telle opération sur telle donnée). Enfin, le SGBD doit également faire respecter la cohérence de sa base de données (crash informatique lors de la création d'un enregistrement par exemple). Afin de communiquer avec le SGBD, l'utilisateur utilise un langage de requêtes. Un SGBD peut-être associé à un ou plusieurs langages de requêtes. Il est courant qu'un SGBD ne communique que via son langage de requête dédié. Prenons pour exemple le langage Cypher qui est dédié à la base de données Neo4j ou encore le langage Cassandra Query Language (CQL) dédié à la base Cassandra.

Une base de données décrit une structure de stockage des données. Il existe plusieurs types de ces structures que l'on appelle des modèles. Le modèle le plus courant est le modèle relationnel (parfois appelé le modèle SQL par abus de langage). Le paragraphe suivant présente un historique des bases de données en prenant soin de présenter chronologiquement les apparitions des différents modèles.

2.3.2.2 Histoire des Bases de Données

En 1956, IBM[14] dévoilait le premier disque dur : le IBM 350[15]. L'invention du disque dur a rapidement remplacé les classiques bandes magnétiques de l'époque et a permis une nouvelle utilisation des ordinateurs : la gestion de fichiers.

[14] https://www.ibm.com/fr-fr/

[15] Plus d'information via https://www-03.ibm.com/ibm/history/exhibits/storage/storage_350.html

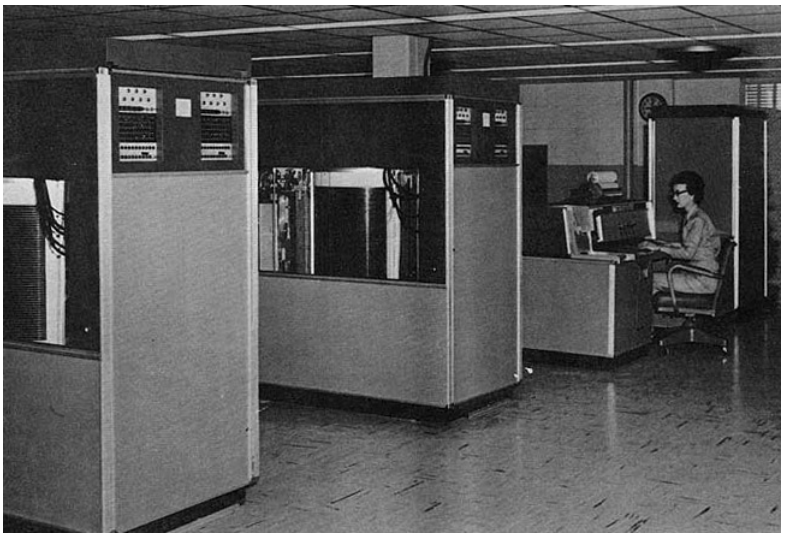


FIGURE 2.21: Photo de deux IBM 350.Source : U. S. Army Red River Arsenal.

Le terme "base de données" est apparu en 1964 dans un journal de recherche et développement de IBM. La première base de données créée suivait le modèle hiérarchique. Quelques années après sa création, Charles Bachman mis en place le modèle réseau. Au tout début des années 1970, Edgar F. Codd publiait (Codd, 1970). Cet article de recherche est l'article fondateur du célèbre modèle relationnel. Il y décrit le modèle de tables, colonnes et lignes que l'industrie utilise dans une majorité de ses applications aujourd'hui encore. En se basant sur ce papier, IBM et l'université de Berkeley ont produit chacun leur système de gestion de base de données : respectivement System-R(Astrahan et al., 1976) et INGRES(Stonebraker et al., 1976). System-R en profite pour être la première base de données à implémenter le langage SQL. INGRES deviendra quant à lui Post-Ingres (Stonebraker and Rowe, 1986) puis le célèbre PostGreSQL (Momjian, 2001) à partir de 1995. Ce modèle qui a la particularité de respecter des propriétés (appelées propriétés ACID) strictes garantissant une transaction au sein d'une base est décrit plus en détail dans le paragraphe suivant.

En 1976, PPS Chen publiait un papier donnant les bases au modèle entité-association (Chen, 1976). Ce modèle permet une représentation graphique haut niveau du modèle de données. Réaliser un modèle entité-association est devenu une étape préliminaire incontournable de la création d'une base de données.

En 1984, Copeland publie un article posant le socle des bases de données orientées objet (Copeland and Maier, 1984). Ce modèle de base a pour objectif d'aller de pair avec les langages orientés objet. Ces langages de programmation tels que C++, Python ou encore le célèbre Java devenant très populaire chez les développeurs dans les années 1990, ce modèle de base de données a connu un certain succès à son époque. Cependant, le modèle

relationnel tient toujours sa place de grand leader. Ainsi pour séduire les utilisateurs de ce dernier, les années 1990 ont aussi vu l'arrivée des bases de données relationnel-objet que M Stonebraker et D Moore ont appelé la "nouvelle grande vague" dans (Stonebraker and Moore, 1995). Cependant, ce modèle n'a pas remporté le succès qu'on lui prédisait.

Les années 2000 sont celles des modèles dits "NoSQL" avec notamment l'arrivée de la base de données orientée graphe Neo4j en 2000 et la base de données orientée colonnes BigTable en 2004. Ces modèles sont souvent le résultats des travaux de recherche des géants du web tels que Google, facebook[16], Yahoo![17], Amazon[18], LinkedIn[19], Twitter[20], etc. Le mouvement NoSQL est présenté plus en détail au paragraphe "Base de données dite NoSQL". Le principe global de ce dernier est de sacrifier les propriétés ACID chères au modèle relationnel dans le but de gagner en performance.

En 2011, Matthew Aslett employait l'expression "NewSQL" dans un rapport de recherche du 451 Group[21]. Dans ce document, il y discute la création d'un modèle relationnel respectant les propriétés ACID mais ayant les performances d'un modèle NoSQL. L'histoire n'accordera que peu d'importance au mot clé "NewSQL", cependant, la prédiction est bonne puisque c'est le respect des propriétés ACID chemin emprunté par les grands noms du NoSQL tel que Neo4j[22].

2.3.2.3 Base de Données Relationnelle

Comme présenté dans le paragraphe précédent, (Codd, 1970) est le papier fondateur du modèle relationnel. Depuis les années 1990, ce modèle est très largement répandu dans les entreprises et dans les salles de classe des universités. Selon ce modèle, les données sont organisées dans des tables. Une table est composée d'une ou plusieurs colonnes (également appelés attributs). Enfin, un tuple est une ligne d'une table et correspond à un enregistrement complet associant chaque attribut de la table à une donnée. Les tables peuvent être liées entre elles via des clés primaires et secondaires. Le langage de requêtes associé au modèle relationnel est le célèbre SQL.

L'intérêt du modèle relationnel repose sur sa capacité à respecter les propriétés ACID. Ces propriétés, si elles sont suivies, permettent de garantir une transaction au sein de la base de données. Ces dernières sont au nombre de quatre :

[16] https://www.facebook.com/
[17] https://fr.yahoo.com/
[18] https://www.amazon.fr/
[19] https://www.linkedin.com/
[20] https://twitter.com/
[21] Plus d'informations sur le rapport via https://451research.com/report-short?entityId=66963
[22] La page https://neo4j.com/developer/graph-database/ explique que "Neo4j provides full database characteristics including ACID transaction compliance".

- L'Atomicité assure qu'une transaction a eu lieu à 100% ou pas du tout,
- la Cohérence assure que la base de données est dans un état valide après une transaction,
- l'Isolation assure que les transactions sont toutes indépendantes et que l'exécution en simultané de plusieurs transactions n'aura aucune conséquence sur l'état du système et des transactions en elles-mêmes,
- la Durabilité assure que les transactions confirmées seront toutes exécutées.

Cependant, les bases de données relationnelles ont un défaut : elles sont tellement exigeantes en termes de sécurité transactionnelle qu'elles en deviennent lentes. Un nombre trop important de transactions mène à la mise hors service du système. En outre, le schéma relationnel est celui des jointures (joindre deux tables entre elles via des clés primaires et secondaires). Or, une jointure est une opération coûteuse en termes de traitement et de mémoire utilisée pour un système. Le faire dans le contexte des mégadonnées ralentit significativement ce dernier. C'est notamment pour cette raison, qu'en parallèle de l'émergence des mégadonnées est apparu une liste de SGBD non relationnels (dits NoSQL) sacrifiant les propriétés ACID pour l'augmentation des performances. (Stonebraker, 2010) propose une comparaison de ces deux grands mouvements du stockage des données.

2.3.2.4 Base de Données NoSQL

Les bases de données NoSQL sont des bases de données non relationnelles. Le terme NoSQL est très controversé. Ils signifient Not Only SQL pour certains (dont la page Wikipédia Française) ou Non SQL pour d'autres (dont la page Wikipédia Anglaise). Quoi qu'il en soit, la communauté s'accorde à lui donner la définition de catégorie regroupant l'ensemble des bases de données autre que hiérarchique, objet ou relationnelle. (Han et al., 2011) propose un panorama des différentes bases NoSQL.

Il existe différents modèles de bases de données dont les principaux sont les suivants :

- Orienté clé/valeur : Les données sont représentées comme une collection de couples clé/valeur. Il s'agit du modèle le plus simple que l'on compare facilement à une table de hachage. Exemples de SGBD : Redis, Riak, Voldemort (LinkedIn), etc.
- Orienté document : Le principe est d'associer à une clé un document regroupant différentes valeurs. Ces documents sont souvent représentés par des fichiers JSON

ou XML (dépend du SGBD). Ce modèle se rapproche beaucoup du modèle clé/-valeur. Exemples de SGBD : MongoDB, couchbase, couchDB, etc.

- Orienté colonnes : Ce modèle peut être vu comme une combinaison des modèles clé/valeur et relationnel. Une base de données orientée colonnes est composée de plusieurs familles de colonnes. Une famille de colonnes regroupe plusieurs colonnes à la manière d'une table pour un modèle relationnel. Cependant, c'est sous forme de clé/valeur que vont être stockées les données, la clé étant le nom de la colonne et la valeur la donnée associée. Exemples de SGBD : Cassandra (Facebook), BigTable (Google), HBase, etc.

- Orienté Graphe : Ce modèle est destiné uniquement aux données ayant une architecture de graphe (données extraites d'un réseau social par exemple). Le modèle graphe permet la représentation sous forme de nœuds et d'arcs. Exemples de SGBD : Neo4j, FlockDB, etc.

La plupart des SGBD cités ci-dessus ont leur propre langage de requêtes. Certains se rapprochent au maximum du langage SQL pour des raisons liées au monopole pendant plus de deux décennies du modèle relationnel. D'autres adaptent le langage de requêtes aux spécificités de leur modèle. C'est le cas, par exemple, du langage Cypher associé à la base de données graphe Neo4j qui représente une requête avec des symboles représentant les éléments d'un graphe.

Le mouvement NoSQL est un mouvement qui suit les besoins des mégadonnées. Ainsi, la plupart des SGBD ont été mis en place avec la forte contrainte du passage à l'échelle (scalable).

2.3.3 Traitement des Mégadonnées

L'analyse de données est un domaine qui date de plusieurs siècles avant l'apparition du premier ordinateur. A l'époque, les données étaient traitées manuellement et conservées dans des livres. La démocratisation des ordinateurs et l'accessibilité du stockage de données ont permis aux logiciels d'analyse de données de voir le jour. Ces logiciels (tels que SPSS, Statistica, HyperCube, SAS ou encore R) ont permis d'accélérer significativement les recherches dans le domaine. Cependant ces logiciels ont une limite forte : ils s'exécutent sur la machine sur laquelle ils sont installés. Ainsi, la capacité de traitements des données dépend essentiellement des propriétés de la machine utilisée. Or, les capacités de stockage et de calcul des machines ne suivent pas la courbe exponentielle de la production des données.

L'arrivée des mégadonnées a bouleversé ce système qui tenait en place depuis plusieurs décennies. Les mégadonnées ont introduit une notion d'échelonnage (passage à l'échelle; scalabilité) qui n'était pas nécessaire dans les anciens systèmes (une machine pouvait gérer à elle seule le petit nombre de données). L'idée de l'échelonnage est de pouvoir passer d'un ensemble de données d'une taille donnée à un autre ensemble de données beaucoup plus gros sans que cela implique un coût financier important. Pour répondre à cette problématique d'échelonnage, il existe une solution : la distribution des données et des calculs au travers d'un ensemble de machines appelé un cluster (aussi appelé grappe). L'objectif est de connecter plusieurs machines entre elles et de les faire traiter chacune une partie de l'ensemble de données. Ainsi, en cas de monter en charge ou d'un arrivage important de nouvelles données, il suffit d'ajouter de nouvelles machines au réseau. Cependant ces clusters posent plusieurs problèmes : Comment ordonnancer efficacement les données et les calculs entre les différentes machines ? Où sont stockées les données ? Comment gérer la sécurité d'un tel système ? Que faire en cas de panne d'une machine ? Etc.

Toutes ces questions ont mené à la création d'outils de traitements de données dans un environnement distribué (stockage distribué et système de calculs distribués). C'est en s'inspirant des publications de Google (Ghemawat et al., 2003), (Dean and Ghemawat, 2008) et (Chang et al., 2008) que Doug Cutting a créé la plateforme Hadoop (White, 2012). Hadoop a été le premier logiciel de traitement distribué de données et reste aujourd'hui le plus suivi et le plus utilisé par la communauté. D'autres logiciels ont suivi : Spark, CMR, HPCC et Hydra. Spark étant le seul logiciel de la liste ayant remporté un certain succès commercial.

Dans les paragraphes suivants, je présente les deux briques essentielles à un système de calculs distribués : le système de fichiers qui permet la gestion des fichiers au sein d'un cluster et la framework de calcul distribué. Puis je détaille les deux principaux logiciels du domaine : Hadoop et Spark.

2.3.3.1 le Système de Fichiers Distribués

Un système de fichiers (abrégé *FS*) a pour rôle de répartir physiquement et organiser les données sur les différents espaces de stockage disponibles. Son objectif est également d'offrir une vue abstraite de cette organisation à l'utilisateur lui permettant de gérer ses fichiers de manière totalement transparente. On parle de système de fichiers distribué lorsque les données sont réparties non pas sur une machine mais plusieurs.

Des systèmes de fichiers distribués comme GoogleFS ou Haddop Distributed File System (HDFS) regroupent les données en blocs de 64 mégaoctets pour éviter un nombre trop

élevé de lecture et écriture sur disque. Ces opérations faisant parties des plus coûteuses dans une application informatique.

2.3.3.2 Framework de Calculs Distribués

La liste des frameworks de calculs distribués n'est pas longue et la plupart ont une base commune : le modèle MapReduce. Ce modèle a été créé par Google et publié en 2008 après quelques années de fonctionnement au sein de l'entreprise. Le nom MapReduce est la contraction des deux méthodes célèbres de la programmation fonctionnelle (telles que Haskell et Scala) : *Map()* et *Reduce()*.

En réalité, lorsqu'on parle de MapReduce, on introduit également deux méthodes non implémentées par l'utilisateur mais par le modèle directement : les méthodes shuffle() et sort(). Ces méthodes sont déclenchées une fois les méthodes map() exécutées et avant l'exécution des méthodes reduce(). Elles ont pour objectif de trier les résultats des méthodes map() et de les répartir dans les différents Reducers (entités exécutant les méthodes reduce()). MapReduce permet de calculer rapidement des données à partir de données indépendantes. L'indépendance des données en entrée est très importante ici.

Le développeur utilisant le modèle MapReduce doit définir le comportement des méthodes map() et reduce(). Ce modèle étant complexe à comprendre si le lecteur n'est pas à l'aise avec la programmation fonctionnelle, je présente son fonctionnement au travers de l'exemple basique du WordCount (comptage de mots). Cet exemple est considéré comme étant le HelloWorld de la programmation fonctionnelle. La figure 2.22[23] présente son schéma de fonctionnement. Les paramètres de cet exemple sont les suivants : les données d'entrée sont un texte et la sortie souhaitée est l'occurrence de chacun des mots (un mot étant identifié comme un groupe de caractères séparés par un espace).

Le principe de fonctionnement est le suivant. La méthode map() divise les données d'entrée et applique sur chacune un traitement spécifique. Pour retourner les mots de notre texte, il est nécessaire de diviser ce dernier en fonction des espaces (on utilise la méthode split() généralement). La méthode map() prend chacun des mots et les traite un par un. Pour notre exemple, nous souhaitons simplement obtenir l'occurrence de chaque mot. L'unique instruction de la méthode map() est donc d'associer la cardinalité 1 à chacun des mots rencontrés. Le retour de la méthode map() est donc une liste de couple de type (clé, valeur) avec pour clé le mot rencontré et pour valeur la cardinalité 1. Le rôle de cette méthode s'arrête ici. Les fonctions shuffle() et sort() prennent en entrée les couples de type (mot, 1) retournés par la méthode map(), trie les résultats par clé et les envoient aux reducers qui appliqueront la fonction reduce(). Cette fonction prend

[23]Source : section Computer Science de l'Université de Calvin.

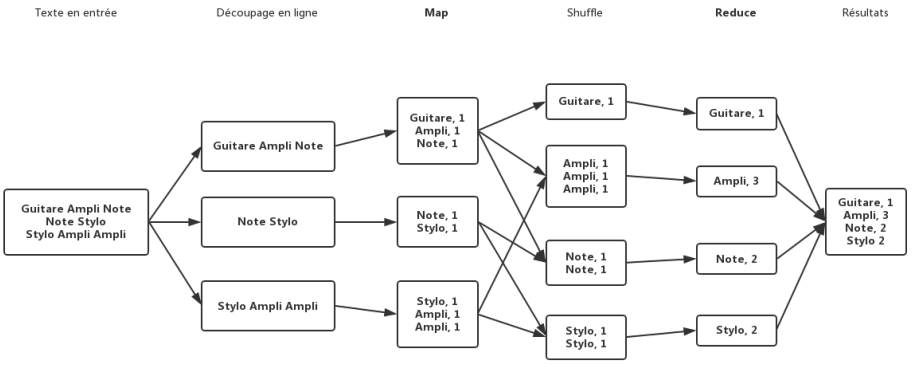


FIGURE 2.22: Exemple du wordCount avec la méthode de MapReduce

en entrée deux couples de type (clé, valeur), y applique un traitement et retourne un nouveau couple si certaines conditions sont satisfaites. Dans notre exemple, la fonction reduce() additionne les cardinalités des couples dont les clés sont identiques. Ainsi, une fois que l'ensemble des opérations de "reduce" sont exécutées, nous devrions retrouver pour chaque mot, sa cardinalité associée.

2.3.3.3 Hadoop

Hadoop est un framework de développement de calculs distribués créé par Doug Cutting suite à trois importantes publications de Google. A l'époque Cutting travaillait pour le compte du moteur de recherche Yahoo!. Il travaillait en collaboration avec Mike Cafarella sur la mise en place d'un nouveau moteur de recherche nommé "Nutch" depuis 2002. Lors de la publication du système de fichiers de Google suivi du moteur MapReduce, Cutting y avoue avoir trouvé une généralisation nécessaire pour son outil Nutch. Il dit notamment : "What they spent a lot of time doing was generalizing this into a framework that automated all these steps that we were doing manually". Suite à cela, Cafarella et Cutting ont créé un système de fichiers distribué ainsi qu'un moteur distribué de type MapReduce afin d'y brancher leur moteur de recherche Nutch et ainsi améliorer les performances de ce dernier. Hadoop a été mis en production le 28 janvier 2006, date considérée par la communauté comme la date de naissance de l'outil. Le nom Hadoop fait référence à la peluche (un éléphant jaune) de l'enfant de Cutting. Plus tard, on trouvera une signification à l'acronyme Hadoop : "High-availability distributed object-oriented platform".

Hadoop était le premier framework open-source dédié au calcul distribué. C'est la raison pour laquelle il a remporté un franc succès. Vite intégré dans la fondation Apache, Hadoop fait désormais partie des logiciels les plus importants de cette dernière (top-level

project). Le projet compte aujourd'hui (mars 2016) plus de 1,7 millions de lignes de code créée par plus de 800 contributeurs au travers de près de 15000 commits[24]. La version actuelle de Hadoop est 2.7.3.

Le cœur de Hadoop est divisé en deux parties. Le système de fichier de Hadoop : HDFS (Hadoop Distributed File System) et le framework de calculs distribués : MapReduce. Depuis la version 2 de Hadoop, YARN fait également partie du cœur de l'application. YARN permet la gestion des ressources au sein d'un cluster. Notons que Hadoop se réfère également à tout un écosystème gravitant autour du cœur de l'application. Dans cet écosystème se retrouvent des systèmes de gestion de la base de données comme HBase (qui est une copie de BigTable proposé par Google). On retrouve aussi des applications permettant le requêtage simplifié avec des outils comme PIG et HIVE, des applications dédiées à la sécurité comme RANGER, des bibliothèques d'algorithmes d'apprentissage automatique comme SAMOA, etc.

2.3.3.4 Spark

Tout comme Hadoop, Spark est un framework de calcul distribué. Il a été développé initialement à l'Université de Berkeley en grande partie par Matei Zaharia en 2009. A partir de 2010, le projet a été rendu open-source et a été intégré par la fondation Apache. A l'heure actuelle, le projet a dépassé les 18 000 commits envoyés par près de 1000 contributeurs. La version actuelle est 2.0.1.

Le projet Spark a démarré suite à l'observation des faiblesses de Hadoop. En effet, Zaharia a observé qu'à chaque job MapReduce, les données étaient lues sur le disque dur et les résultats écrits également sur le disque, créant un ralentissement significatif de l'application dans sa globalité. Suite à cette observation, Zaharia a proposé un système similaire à Hadoop mais utilisant la mémoire cache des machines afin de réduire les interactions avec le disque. Un utilisateur de Spark peut ainsi "monter en mémoire" des données qu'il compte utiliser fréquemment. Avec cette amélioration, Spark est annoncé comme étant jusqu'à 100 fois plus rapide que Hadoop. Notons que Spark peut fonctionner seul (mode standalone) ou en collaboration avec Hadoop et notamment son système de fichiers distribués.

D'un point de vue technique, les collections de données distribuables ont un type dédié : Resilient Distributed Dataset (RDD). Zaharia présente cette nouvelle abstraction dans (Zaharia et al., 2012). Un RDD a plusieurs propriétés :

[24]Retrouvez la page gitHub de Hadoop via `https://github.com/apache/hadoop`

- Les RDDs sont tolérants aux pannes. Si un RDD est perdu (crash d'une machine stockant le RDD), il peut être reconstruit. Cela implique qu'un RDD ne peut être modifié. Modifier un RDD signifie construire un nouveau RDD.
- Les RDDs peuvent être persistés en mémoire. Par exemple, il est courant de persister un résultat intermédiaire car il a vocation à être réutilisé.
- Le partitionnement des RDDs peut être automatique ou manuel, optimisant ainsi le stockage sur les différentes machines du cluster.
- Les RDDs sont manipulables au travers d'une API dédiée.

Au-delà de cette particularité qui fait de Spark l'un des outils les plus rapides de calculs distribués, le logiciel propose différentes bibliothèques permettant la manipulation des données. Il existe deux librairies d'algorithmes d'apprentissage automatique (MLLib et sparkml), une librairie permettant la manipulation et le calcul distribuée de graphe (GraphX), une librairie de traitement distribué en streaming (Spark Streaming) et une librairie permettant de requêter les données à l'aide d'un langage proche du SQL (Spark SQL).

2.3.3.5 Autres Frameworks

Hadoop et Spark sont les deux frameworks leaders du calcul distribué, cependant, d'autres frameworks existent. Je peux citer Cluster Map Reduce (CMR)[25] produit par Chikita, High Performance Computing Cluster (HPCC)[26] ou encore Hydra[27]. Ces logiciels ne jouissent pas d'un engouement médiatique mais sont à surveiller malgré tout.

[25] https://github.com/chitika/cmr
[26] https://hpccsystems.com/
[27] https://github.com/addthis/hydra

Chapitre 3

Classification de Textes Représentés par des Concepts

3.1 Introduction

L'ontologie informatique a pour vocation d'être interprétée par la machine, lui donnant un moyen de lier des connaissances et de faire des interprétations. C'est pour cette raison qu'elles sont encodées à l'aide d'un langage de spécification comme OWL et RDFS. Dans le domaine de la classification de textes, chaque texte est habituellement représenté par une liste de mots (BOW) en vue de les classifier ensuite. Cependant (Bloehdorn and Hotho, 2004) a identifié quatre limites à cette représentation : certains mots sont polysémiques ("orange" se réfère à la couleur, le fruit, l'opérateur téléphonique ou encore la ville du sud de la France), d'autres peuvent être des synonymes et être malgré tout différenciés dans l'analyse ("plectre" a le même sens que "médiator"), d'autres encore sont fortement liés sémantiquement sans que cela soit pris en compte dans la représentation ("chevalet" est très lié à "frette") et enfin, certains mots perdent leur sens s'ils sont extraits de leur groupe nominal (exemple : "poste de police" n'a pas le même sens que "poste" et "police" pris séparément). A ces limites s'ajoute celle du nombre trop importants de mots représentant les documents qu'on appelle la malédiction de la dimension. (Beyer et al., 1999) a montré qu'à partir d'un nombre suffisamment important de dimensions, les observations (des textes dans notre étude) deviennent équidistants et sont donc difficilement comparables. En réponses à ces limites, des techniques permettent de représenter les textes par des concepts d'une ontologie (BOC), intégrant la notion de sens à l'analyse (Sahlgren and Cöster, 2004) (Kehagias et al., 2003).

Dans ce chapitre, je présente la méthode utilisée pour construire le BOC d'un texte et son intégration au sein du modèle de classification. Le modèle ainsi créé sert de base à

mes contributions présentées dans les chapitres 4 et 5. J'expose également les données utilisées et leurs prétraitements associés pour construire le modèle.

Dans une première partie, je présente le corpus de textes et l'ontologie utilisés pour la constitution du modèle. Dans une seconde partie, j'expose les étapes nécessaires à la représentation des textes par des concepts. Dans une troisième partie, je présente l'algorithme de classification employé et je termine avec une quatrième partie en présentant une visualisation interactive du modèle.

3.2 Ensembles de Données

Les corpus de textes labellisés permettent la construction et l'évaluation des méthodes de classification. Différents corpus sont utilisés par la communauté dont les principaux sont présentés dans la section 2.2.7. Les méthodes représentant les textes avec des concepts extraits d'ontologie intègrent une seconde donnée d'entrée : l'ontologie de domaine. Pour fonctionner ensemble, le corpus et l'ontologie doivent partager le même domaine de connaissances. Dans cette section, je présente les choix des données d'entrée : l'ontologie *Disease Ontology* (DO) et le corpus *Ohsumed*.

3.2.1 L'Ontologie Disease Ontology

Pour mener mes différentes expériences de classification de textes représentés par des concepts, j'ai fait le choix d'utiliser l'ontologie de domaine *Disease Ontology* (Schriml et al., 2012) que je note DO. Ci-après, j'explique les raisons de ce choix et présente en détail celle-ci.

Comme Gruber l'indique, une ontologie est nécessairement encodée dans un format interprétable par une machine. Dans la section 2.1.2, j'ai présenté les différents langages de spécification. Parmi ceux-ci, le langage recommandé par l'organisme du W3C est OWL. Pour assurer la pérennité de mes travaux, j'ai fait le choix de travailler avec une ontologie écrite dans ce langage écartant notamment l'ontologie MeSH (la version OWL de cette dernière n'est pas officielle) et les différentes applications sémantiques orientées WEB.

En outre, le choix de l'ontologie doit se faire conjointement avec celui du corpus de textes. Les recherches dans le domaine médical sont très actives et ont mené à la création d'ontologies dont (Yao et al., 2011) propose une étude comparative et de corpus de textes labellisés (Ohsumed).

DO est une ontologie traitant du domaine des maladies humaines, elle est rédigée dans la langue anglaise. Elle a été créée et est encore maintenue par les chercheurs de l'université de Northwestern, du centre de la médecine génétique et de l'école de médecine de l'université du Maryland. L'ontologie intègre le vocabulaire des maladies de *Medical Subject Headings (MeSH)* (Lipscomb, 2000), *International Classification of Diseases* (ICD) (Fritz et al., 2000), le thésaurus du *National Cancer Institute* (NCI) (Golbeck et al., 2003), SNOMED (Spackman et al., 1997) et de *Online Mendelian Inheritance* in Man (OMIM) (Hamosh et al., 2005).

DO contient 9247 concepts et 7082 relations sémantiques. Je note $E = \{e_1, ..., e_l\}$ l'ensemble des l concepts de l'ontologie et $d_{(e_i,e_j)}$) le nombre minimal de sauts nécessaires pour relier les concepts e_i et e_j.

Chaque concept est défini par une classe, est associé à un label (rdfs:label) et contient 0 ou n synonymes (oboInOwl:hasExactSynonym). La figure 3.1 présente la description du concept dont l'identifiant est $DOID3078$. On observe que le label associé à ce concept est *anaplastic childhood astrocytoma* et que deux synonymes (*Undifferentiated pediatric astrocytoma* et *pediatric Glioblastoma Multiforme*) ont été définis. Les relations sémantiques sont exclusivement des relations hiérarchiques d'hyperonymie et d'hyponymie. Par exemple, on observe sur les figures 3.1 et 3.2 que le concept $DOID3078$ est une sous classe (rdfs:subClassOf) du concept $DOID3069$. Autrement dit, le concept $DOID3069$ est l'hyperonyme du concept $DOID3078$ et le concept $DOID3078$ est l'hyponyme du concept $DOID3069$.

Enfin, il est possible de naviguer dans l'ontologie à l'aide de l'outil de visualisation mis à disposition par Université du Maryland[1]. Le conteneur à gauche de la page est un menu déroulant qui permet de naviguer dans l'ontologie sous la forme de dossiers/sous-dossiers. Le conteneur principal permet d'afficher les informations d'un concept (figure 3.2) et de visualiser de manière interactive l'ontologie (figure 2.1). Enfin une barre de recherche simple et une recherche avancée est proposée à l'utilisateur.

3.2.2 Corpus de Textes Ohsumed

Parmi les corpus de textes labellisés listés dans la section 2.2.7, seul Ohsumed est spécifique à un domaine de connaissances. Tout comme *DO*, il traite des maladies humaines. C'est principalement pour cette raison que mon choix s'est orienté vers ce dataset. Ci-après, je présente dans le détail Ohsumed et expose les différents prétraitements réalisés pour l'adapter à la construction du modèle de classification.

[1]Disponible sur la page d'accueil de `http://disease-ontology.org/`.

```
<!-- http://purl.obolibrary.org/obo/DOID_3078 -->

<owl:Class rdf:about="http://purl.obolibrary.org/obo/DOID_3078">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        grade III astrocytoma
    </rdfs:label>
    <rdfs:subClassOf rdf:resource="http://purl.obolibrary.org/obo/DOID_3069"/>
    <obo:IAO_0000115 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        An astrocytoma that is characterized by cells with regular, round to oval nuclei.
    </obo:IAO_0000115>
    <oboInOwl:id rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        DOID:3078
    </oboInOwl:id>
    <oboInOwl:hasDbXref rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        MSH:D001254
    </oboInOwl:hasDbXref>
    <oboInOwl:hasDbXref rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        NCI:C9477
    </oboInOwl:hasDbXref>
    <oboInOwl:hasDbXref rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        SNOMEDCT_US_2015_03_01:55353007
    </oboInOwl:hasDbXref>
    <oboInOwl:hasDbXref rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        UMLS_CUI:C0334579
    </oboInOwl:hasDbXref>
    <oboInOwl:hasExactSynonym rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        anaplastic astrocytoma
    </oboInOwl:hasExactSynonym>
    <oboInOwl:hasOBONamespace rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        disease_ontology
    </oboInOwl:hasOBONamespace>
    <oboInOwl:hasExactSynonym rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        grade III Astrocytic tumor
    </oboInOwl:hasExactSynonym>
    <oboInOwl:inSubset rdf:resource="http://purl.obolibrary.org/obo/doid#DO_cancer_slim"/>
</owl:Class>
```

FIGURE 3.1: Définition du concept *DOID*3077 de DO.

Metadata Visualize

DOID	DOID:3078
Name	grade III astrocytoma
Definition	An astrocytoma that is characterized by cells with regular, round to oval nuclei. http://en.wikipedia.org/wiki/Anaplastic_astrocytoma, http://www.abta.org/brain-tumor-information/types-of-tumors/astrocytoma.html, http://www.cancer.gov/dictionary?CdrID=45591
Xrefs	MSH:D001254 NCI:C9477 SNOMEDCT_US_2016_03_01:55353007 UMLS_CUI:C0334579
Subsets	DO_cancer_slim
Synonyms	anaplastic astrocytoma [EXACT] grade III Astrocytic tumor [EXACT]
Relationships	is_a astrocytoma

Add an item to the term tracker

FIGURE 3.2: Définition du concept *DOID*3078 de DO.

Le corpus original de Ohsumed regroupe un total de 56984 abstracts médicaux datant de l'année 1991 extraits de la base MEDLINE. Dans son étude (Joachims, 1998), Joachims a proposé une version raccourcie de 20000 textes. Les 10000 premiers textes sont utiles à l'apprentissage de son modèle et les 10000 suivant à l'évaluation de ce dernier. Cet ensemble de données a été repris ensuite pour différentes recherches telles que (Moschitti and Basili, 2004) et (Díaz et al., 2004). Dans sa version originale et raccourcie, les textes sont regroupés dans les 23 catégories du groupe "Maladies" du thesaurus *Medical Subject Headings (MeSH)*. Ci-après, une courte description de chacune des catégories (l'identifiant de la catégorie est donnée entre crochets et le nom complet en gras) :

- **[C01] : Infections bactériennes et mycoses**. Infections causées par des bactéries et des champignons, générales, spécifiées ou non précisées.
- **[C02] : Maladies virales**. Terme générique pour les maladies produites par les virus.
- **[C03] : Maladies parasitaires**. Infections ou infestations par des organismes parasites. Ils sont souvent contractés par contact avec un vecteur intermédiaire, mais peuvent se produire à la suite d'une exposition directe.
- **[C04] : Néoplasmes**. Nouvelle croissance anormale des tissus. Les néoplasmes malins présentent un degré plus élevé d'anaplasie et ont les propriétés d'invasion et de métastase, par rapport aux néoplasmes bénins.
- **[C05] : Maladies musculo-squelettiques**. Affection nocive et douloureuse causée par une utilisation excessive ou une surexcitation d'une partie du système musculo-squelettique, résultant souvent d'activités physiques liées au travail. Elle est caractérisée par une inflammation, une douleur ou un dysfonctionnement des articulations, des os, des ligaments et des nerfs.
- **[C06] : Maladies du système digestif**. Maladies de toute partie de l'appareil digestif ou des organes accessoires (fois, voies Biliaires, Pancréas).
- **[C07] : Maladies stomatiques**. Maladies générales ou non spécifiées du système stomatognathique, comprenant la bouche, les dents, les mâchoires et le pharynx.
- **[C08] : Maladies respiratoires**. Maladie de l'appareil respiratoire ou provoquant des troubles de la respiration.
- **[C09] : Maladies oto-rhino-laryngologiques**. Processus pathologiques de l'oreille, du nez et de la gorge, également connus sous le nom de maladies ORL.

- **[C10] : Maladies du système nerveux**. Maladies du système nerveux central et périphérique. Cela comprend les troubles du cerveau, de la moelle épinière, des nerfs crâniens, des nerfs périphériques, des racines nerveuses, du système nerveux autonome, de la jonction neuromusculaire et du muscle.
- **[C11] : Maladies des yeux**. Maladies affectant l'œil.
- **[C12] : Urologie et Maladies génitales masculines**. Processus pathologiques de l'appareil uro-génital masculin et du système reproducteur.
- **[C13] : Maladies génitales féminines et complications de la grossesse**. Processus pathologiques de l'appareil uro-génital féminin, le système reproducteur, et les troubles liés à la grossesse.
- **[C14] : Maladies cardiovasculaires**. Les états pathologiques impliquant le système cardio-vasculaire y compris le cœur; les Vaisseaux sanguins; ou le pericardium.
- **[C15] : Maladies hépatiques et lymphatiques**. Les maladies hématologiques et les maladies du système lymphatique collectivement. Les maladies hémiques comprennent des troubles impliquant les éléments formés et des composants chimiques; Les maladies lymphatiques comprennent les troubles liés à la lymphe, aux ganglions lymphatiques et aux lymphocytes.
- **[C16] : Maladies néonatales et anomalies**. Maladies existant à la naissance et souvent avant la naissance, ou qui se développent pendant le premier mois de vie, indépendamment de la causalité. Parmi ces maladies, celles caractérisées par des déformations structurales sont appelées anomalies congénitales.
- **[C17] : Maladies de la peau et des tissus conjonctifs**. Un terme collectif pour les maladies de la peau et de ses appendices et du tissu conjonctif.
- **[C18] : Maladies nutritionnelles et métaboliques**. Terme collectif désignant les troubles nutritionnels résultant d'une mauvaise absorption ou d'un déséquilibre nutritionnel et de troubles métaboliques résultant de défauts de biosynthèse (anabolisme) ou de dégradation (catabolisme) de substances endogènes.
- **[C19] : Maladies endocriniennes**. Les processus pathologiques des glandes endocrines, et les maladies résultant du niveau anormal des hormones disponibles.
- **[C20] : Maladies immunologiques**. Troubles causés par des mécanismes immunologiques anormaux ou absents, humoraux ou cellulaires, ou les deux.
- **[C21] : Troubles d'origine environnementale**. Troubles causés par des forces externes plutôt que par une dysfonction physiologique ou par des agents pathogènes.

Increased serotoninergic and noradrenergic activity in hepatic encephalopathy in rats with thioacetamide-induced acute liver failure.
Functional changes of various neurotransmitter systems have been implicated in the pathogenesis of hepatic encephalopathy.
In this study the role of brain monoaminergic neurotransmitter systems in hepatic encephalopathy was investigated in rats with thioacetamide-induced acute liver failure.
Concentrations of serotonin, dopamine, noradrenaline and of their metabolites 5-hydroxyindoleacetic acid, dihydroxyphenylalanine (following inhibition of dihydroxyphenylalanine-decarboxylase), dihydroxyphenylacetic acid, homovanillic acid and 3-methoxy-4-hydroxyphenyl-glycol, were measured in the cerebral cortex, striatum and hippocampus by high performance liquid chromatography with electrochemical detection.
In hepatic encephalopathy concentrations of 5-hydroxyindoleacetic acid were increased in all three brain areas (196%, 204% and 264% of saline-treated controls, p less than 0.01), and concentrations of serotonin were increased in the frontal cortex (121%, p less than 0.01).
In the frontal cortex and hippocampus of encephalopathic rats dopamine levels were increased (157% and 289%, p less than 0.05), and levels of noradrenaline (53% and 46%, p less than 0.05) were decreased associated with increased 3-methoxy-4-hydroxyphenylglycol levels (173% and 206%, p less than 0.05).
The extent of these changes correlated with the stage of hepatic encephalopathy.
In hepatic encephalopathy dihydroxyphenylalanine accumulation was increased in the hippocampus and unchanged in the cerebral cortex.
Dopamine, noradrenaline, dihydroxyphenylacetic acid and homovanillic acid concentrations were unchanged in the striatum.
The results of this study indicate that hepatic encephalopathy in thioacetamide-induced acute liver failure in rats is associated with neurochemical changes, suggesting an increased activity of the noradrenergic and serotoninergic neurotransmitter systems.

FIGURE 3.3: Contenu du fichier 0000682 de Ohsumed.

- **[C22] : Maladies animales**. Maladies qui surviennent chez les animaux vertébrés.
- **[C23] : Conditions pathologiques, signes et symptômes**. Conditions anormales anatomiques ou physiologiques et manifestations objectives ou subjectives de la maladie, non classées comme maladie ou syndrome.

Techniquement, les textes sont bruts et encodés en UTF-8, facilitant les travaux d'analyse automatique du langage. Chaque phrase est retournée à la ligne et la première correspond au titre de l'abstract. A partir de la seconde ligne, toutes les phrases commencent par un espace. Enfin, la dernière ligne est systématiquement vide. Les textes contiennent environ 10 lignes et le nombre de mots par ligne est variable (certaines pouvant contenir 5 mots et d'autres plus de 50. La taille des textes est ainsi très variable. Les textes sont écrits en anglais et nombre d'entre eux sont accompagnés de caractères spéciaux tels que

"%", "?" et nombreux autres. Chaque texte est contenu dans un fichier sans extension. Le nom de ces fichiers est un nombre à sept chiffres (commençant par autant de 0 que nécessaire). Par exemple, la figure 3.3 présente le texte du fichier 0000682 labellisé par la catégorie $C10$. Ce dernier contient un total de 11 lignes (en comptant la dernière ligne vide) pour un total de 249 mots.

L'ensemble des textes sont répartis dans 23 dossiers, chacun représentant une catégorie. Chaque abstract du corpus peut appartenir à une ou plusieurs catégories faisant de Ohsumed un ensemble de textes dit multi-labels. De ce fait, de nombreux fichiers sont dupliqués dans les différents dossiers. Je note le corpus original $Ohsumed - O$.

Mes travaux se concentrent sur la classification mono-label (un texte n'est associé qu'à une unique catégorie) impliquant la suppression des textes classés dans au moins deux catégories. Le tableau 3.4 donne la répartition originale des textes par catégorie et la répartition après suppression des doublons. Je note $Ohsumed - D$ le sous-ensemble de texte obtenu. On observe que la suppression des doublons a nécessité la suppression de plus de 67% des fichiers (il ne s'agit pas de 67% des textes mais de 67% de l'ensemble des textes duplicata compris).

Le tableau 3.4 et la figure 3.5 mettent en avant un problème récurrent des corpus de textes labellisés dont Ohsumed. En effet, on y observe que les textes ne sont pas répartis équitablement parmi les catégories. La classe la plus petite regroupe 427 textes (catégorie $C03$) alors que la classe la plus peuplée en regroupe 9611 (catégorie $C23$). Soit un écart type total de plus de 2225. La suppression des doublons permet de réduire l'écart type à un peu plus de 753. La catégorie la plus impactée par l'exclusion des doublons est la catégorie $C23$ avec la suppression de 7687 textes. Cela s'explique par la nature de cette dernière. Effectivement, elle regroupe les abstracts traitant des conditions pathologiques, signes et symptômes et non d'un type de maladie en particulier comme les 22 autres.

L'objectif de ce chapitre est la construction d'un modèle de classification de textes représentés par des concepts or, certains textes du sous-ensemble $Ohsumed - D$ peuvent ne pas en contenir. Un concept est associé au texte si et seulement si son label ou un synonyme correspond à un groupe de mots extrait de ce dernier (cette partie est décrite dans la section 3.3.2). Dans le cas où aucun concept n'est associé, le texte n'est attaché à aucune caractéristique. Pour éviter ce cas de figure et se concentrer sur les méthodes de classification, j'ai pris l'initiative de supprimer l'ensemble des textes ne contenant aucun concept de l'ontologie DO. Je note $Ohsumed - D\&C$ le nouveau corpus de textes. Le tableau 3.7 présente les répartitions des classes de $Ohsumed - D$ et $Ohsumed - D\&C$ et la figure 3.6 une comparaison entre ces deux dernières.

Catégories	**Nombre de textes de** $Ohsumed - O$	**Nombre de textes de** $Ohsumed - D$
C01	2540	631
C02	1171	249
C03	427	183
C04	6327	2513
C05	1678	505
C06	2989	837
C07	526	132
C08	2589	634
C09	715	169
C10	3851	1328
C11	998	337
C12	2518	842
C13	1623	473
C14	6102	2876
C15	1277	307
C16	1086	356
C17	1617	592
C18	1919	815
C19	865	200
C20	3116	1060
C21	2933	1283
C22	506	56
C23	9611	1924
Total	56984	18302

FIGURE 3.4: Répartition des textes de $Ohsumed - O$ et $Ohsumed - D$.

On observe que le passage de $Ohsumed - D$ à $Ohsumed - D\&C$ a réduit le nombre de textes de 6855 soit une réduction de plus d'un tiers (37%). L'écart type de la répartition des textes dans les catégories a également diminué passant de 753 à 561.

3.3 Représentation des Textes

La représentation des textes est une étape clef de la classification de textes. La section 2.2.4 présente les différentes méthodes de représentation : en sac de mots, en sac de n-grams, en sac de groupes de mots, en sac de concepts et sac hybride. Mes recherches se focalisent sur l'amélioration des performances de modèles de classification utilisant la représentation des textes par des concepts.

Dans cette section, je présente les différentes techniques employées pour représenter les textes à l'aide de concepts. Dans une première sous partie, je présente l'extraction des groupes de mots ("phrases" en anglais) contenus dans le texte puis j'explique l'association

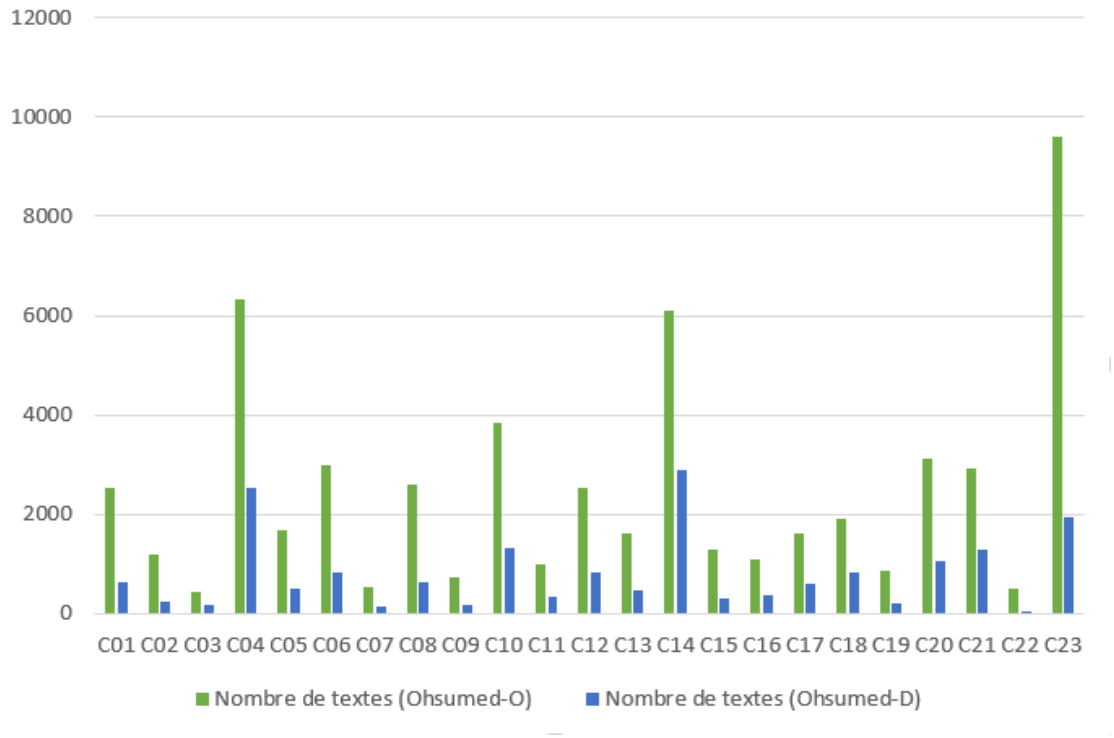


FIGURE 3.5: Graphique en barres de la répartition des textes de $Ohsumed - O$ et $Ohsumed - D$.

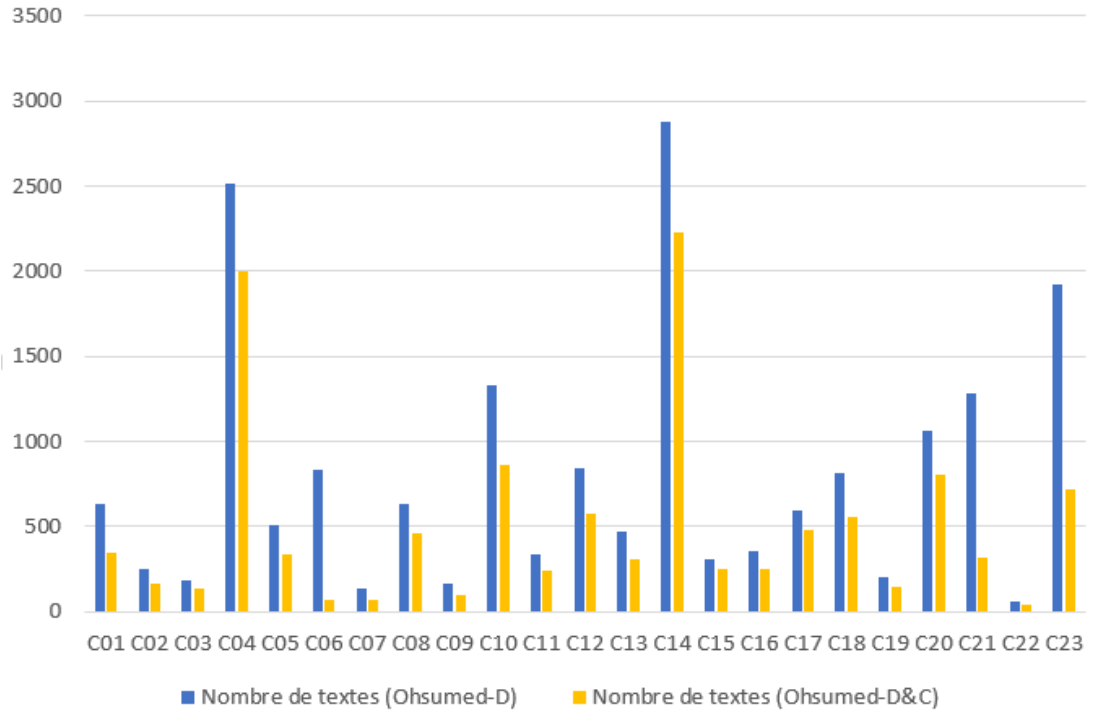


FIGURE 3.6: Graphique en barres de la répartition des textes de $Ohsumed - D$ et $Ohsumed - D\&C$.

Catégories	**Nombre de textes de** $Ohsumed - D$	**Nombre de textes de** $Ohsumed - D\&C$
C01	631	347
C02	249	169
C03	183	132
C04	2513	2002
C05	505	341
C06	837	73
C07	132	65
C08	634	456
C09	169	99
C10	1328	859
C11	337	237
C12	842	572
C13	473	305
C14	2876	2224
C15	307	248
C16	356	249
C17	592	479
C18	815	559
C19	200	149
C20	1060	803
C21	1283	320
C22	56	38
C23	1924	721
Total	18302	11447

FIGURE 3.7: Répartition des textes de $Ohsumed - D$ et $Ohsumed - D\&C$.

de ces derniers avec les concepts de l'ontologie dans une seconde sous partie. Chaque étape est illustrée par un exemple.

3.3.1 Extraction des Mots et des Groupes de Mots

La première étape de la représentation des textes par des concepts consiste à extraire l'ensemble des groupes nominaux contenus dans les textes dans le but de les comparer ensuite aux concepts. Deux familles d'extraction se distinguent (la section 2.2.4.3 développe plus en détail ces deux familles de méthodes) :

- Les méthodes issues des statistiques. Celles-ci visent à assembler des mots simples qui co-occurrent de manière récurrente. Le groupe de mots n'a pas forcément de sens (exemple : *avion pilote*).
- Les méthodes issues du traitement automatique du langage. Celles-ci visent à extraire les groupes de mots à partir de règles de syntaxes. Chaque groupe de

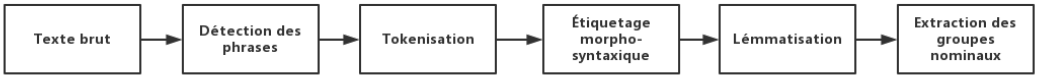


FIGURE 3.8: Workflow des techniques d'extraction des groupes nominaux.

"Cerebellopontine angle lipoma in a teenager.
Lipomas of the cerebellopontine angle are very rare lesions.
To date, 18 patients have been reported, 17 of whom were adults.
A second child is described with cerebellopontine angle lipoma."

FIGURE 3.9: Contenu du texte 0000014 de Ohsumed.

mots extraits a un sens et est contenu dans le texte (exemple : *pilote d'avion*). Cependant, ces méthodes sont gourmandes en temps de calcul.

J'ai fait le choix de baser l'extraction des groupes de mots sur les techniques du traitement automatique du langage. En effet, pour représenter un texte avec des concepts il est nécessaire d'extraire les mots et groupes de mots ayant un sens afin de les comparer avec les synonymes et labels des concepts.

Le déroulement de l'extraction des mots et groupes de mots comprend 5 étapes comme l'illustre la figure 3.8). Le texte 0000014 labellisé par $C04$ de Ohsumed (voir la figure 3.9) me sert de support d'exemple pour présenter ces étapes :

- Détection des phrases avec des expressions irrégulières (voir la figure 3.10). Cette étape permet l'extraction des phrases afin de les traiter chacune de manière indépendante. En cas d'erreur sur une phrase (caractère spécial non pris en compte par exemple), l'ensemble du texte n'est pas impacté.
- Découpages des phrases en unités lexicales appelés tokens (voir la figure 3.11). Dans mes travaux, une unité lexicale est un mot ou un caractère de ponctuation.
- Étiquetage morpho-syntaxique aussi appelé POS tagging (voir la figure 3.12). A cette étape, chaque unité lexicale est associée à une étiquette indiquant sa classe grammaticale.

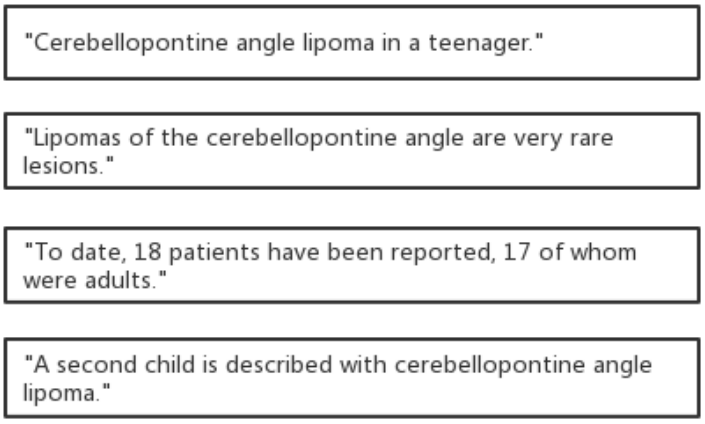


FIGURE 3.10: Étape de découpage du texte en phrases indépendantes.

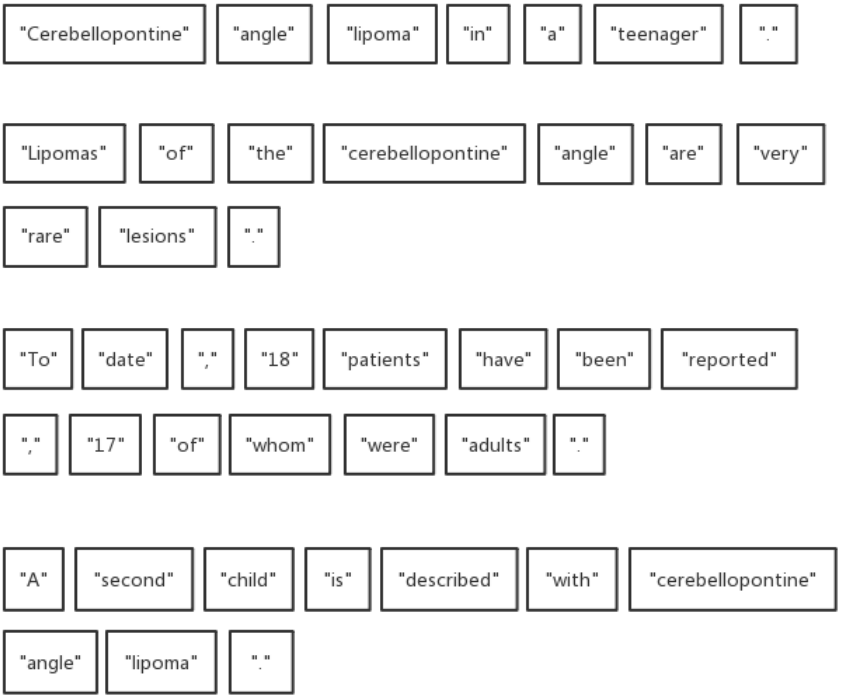


FIGURE 3.11: Étape de découpage des phrases en unités lexicales.

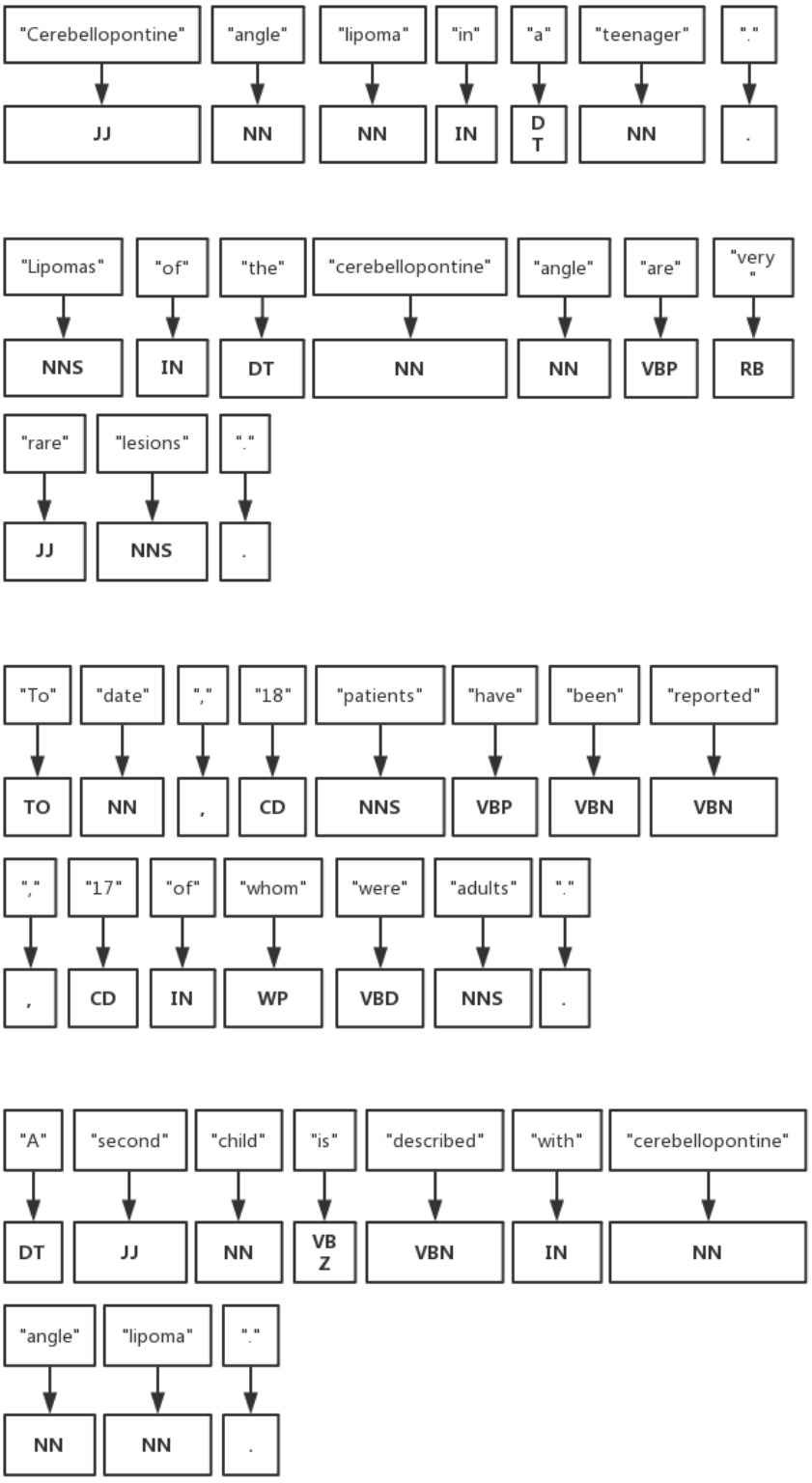


FIGURE 3.12: Étiquetage morpho-syntaxique des unités lexicales.

- Lémmatisation (voir la figure 3.13). Les lemmes de chaque unité lexicale sont retournés permettant d'unifier les caractéristiques. Ainsi, les mots aux pluriels sont transformés au singulier, les verbes passés à l'infinitif, etc.

- Extraction des groupes nominaux (voir la figure 3.14). Dans mes travaux, je n'extrais que les groupes nominaux. En effet, les descripteurs de l'ontologie Ohsumed sont exclusivement des groupes nominaux.

A la sortie de ces cinq étapes, le texte analysé est décrit par une liste de groupes nominaux. Concernant le texte présenté dans l'exemple, les groupes nominaux le décrivant sont : "lipoma", "angle lipoma", "cerebellopontine angle lipoma", "teenager", "angle", "cerebellopontine angle", "lesion", "rare lesion", "date", "patient", "adult", "child" et "second child".

3.3.2 Association des Concepts

L'extraction des mots et groupes de mots est une étape essentielle à la découverte de concepts. En effet, comme je l'ai expliqué dans la section 3.2.1, un concept est décrit par un label et une liste de synonymes. Ces derniers sont des mots ou des groupes de mots. Par exemple, le concept $DOID3078$ (figure 3.1) a pour label un groupe nominal composé de trois mots (*anaplastic childhood astrocytoma*) et deux synonymes de la même forme (*Undifferentiated pediatric astrocytoma* et *pediatric Glioblastoma Multiforme*). Pour simplifier la suite du document, je nomme les *descripteurs* l'ensemble des synonymes et du label d'un concept.

Identifier ces descripteurs peut être réalisé de différentes façons :

- Comparaison avec un extrait du groupe nominal. Cette technique propose qu'un concept soit retourné si la totalité ou un extrait d'un de ses descripteurs est découvert dans le texte. Par exemple, si le mot "childhood" est contenu dans un texte, alors le concept $DOID3078$ y sera associé car son label contient ce mot (*anaplastic* ***childhood*** *astrocytoma*). Cette méthode est permissive et permet l'extraction de nombreux concepts, cependant ces derniers ne représentent pas exactement le contenu du texte et le sens de celui-ci n'est plus respecté.

- Comparaison parfaite. Pour cette technique, l'extraction d'un concept implique que l'un de ses descripteurs est directement et entièrement contenu dans le texte. Ainsi, l'association du concept $DOID3078$ à un texte implique que le dit texte contient au moins l'un des trois groupes nominaux *anaplastic childhood astrocytoma*, *Undifferentiated pediatric astrocytoma* et *pediatric Glioblastoma Multiforme*. Cette

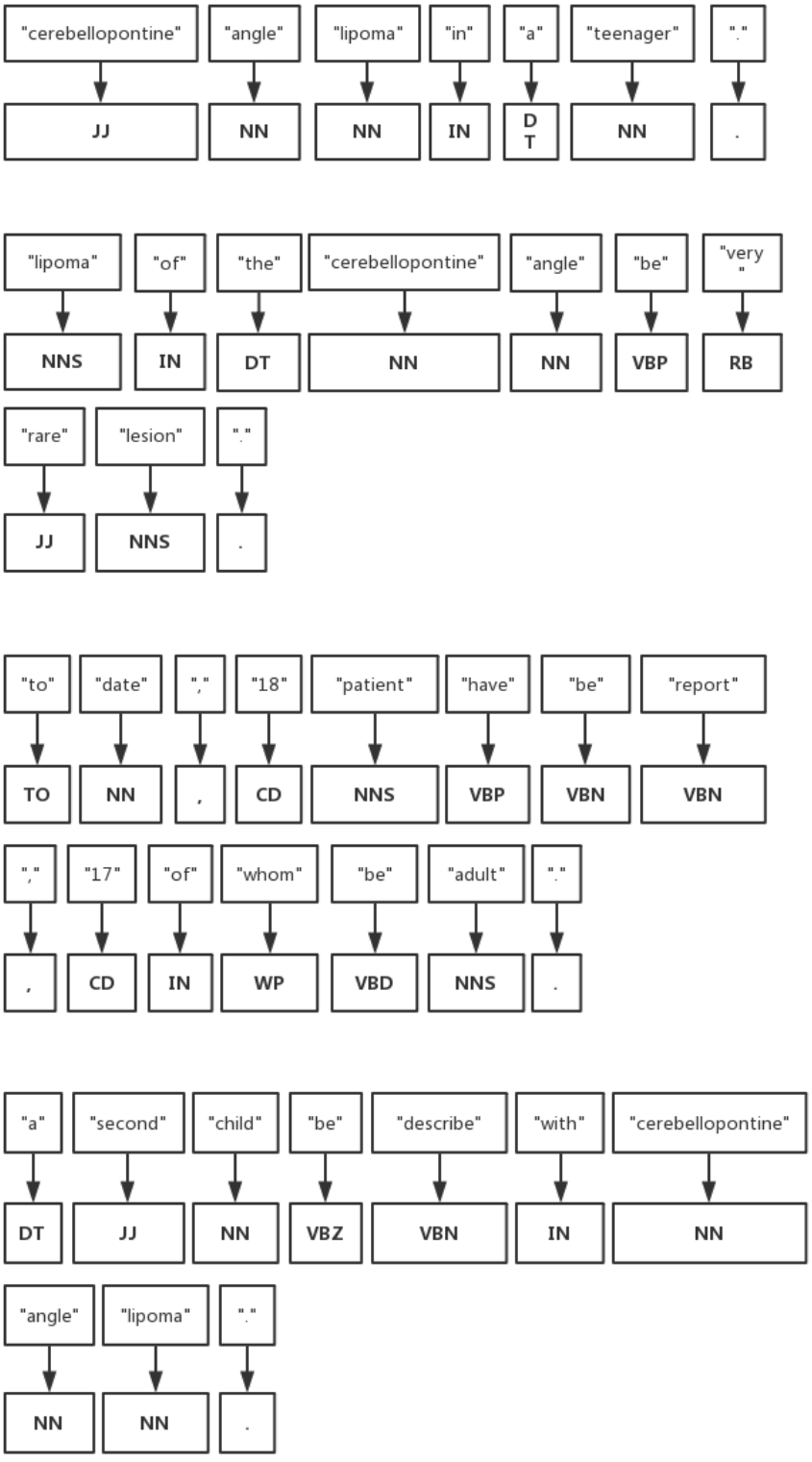


FIGURE 3.13: Lémmatisation des unités lexicales

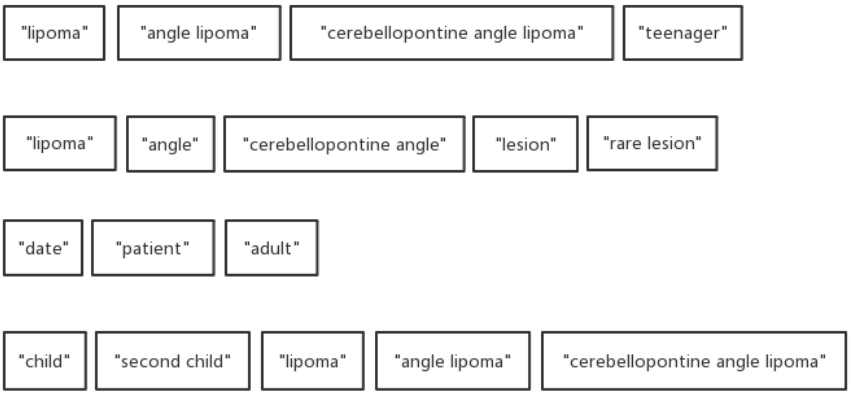


FIGURE 3.14: Extraction des groupes nominaux.

méthode permet de s'assurer que l'ensemble des concepts associés aux documents le sont réellement. L'inconvénient est que les textes sont décrits par peu de concepts.

J'ai fait le choix d'utiliser la méthode de la comparaison parfaite dans le but de conserver une cohérence sémantique de l'analyse. En effet, la première méthode peut introduire du bruit dans la représentation du texte. Parmi les résumés médicaux de Ohsumed, il est courant de rencontrer des mots tels que *adult* ou *child*. Le mot *child* peut être associé à un groupe nominal tel que *childhood leukemia*, *childhood medulloblastoma* ou d'autres. Alors que la seconde méthode de comparaison permet l'extraction exclusive du concept relatif à la leucémie infantile, la première propose un total de 82 concepts contenant le mot *child* dans l'un de leurs descripteurs, ajoutant 81 concepts de bruit à la représentation du texte.

En appliquant la méthode d'extraction sélectionnée à l'exemple de la section 3.3.1, seul le concept ayant pour label *lipoma* est retourné pour représenter le texte. La figure 3.15 le processus de filtrage des groupes nominaux avec les concepts.

A la sortie de cette étape, la représentation finale du texte 0000014 est obtenue. Un unique concept décrit ce dernier : le concept $DOID$: 3315 décrivant le lipoma.

3.4 Classification Naïve Bayésienne

Pour la construction du modèle de classification, j'ai sélectionné l'algorithme du classifieur naïf Bayésien pour sa simplicité, sa facilité de représentation et sa large utilisation dans la communauté (dont entre autres (Peng and Schuurmans, 2003), (Dai et al., 2007)

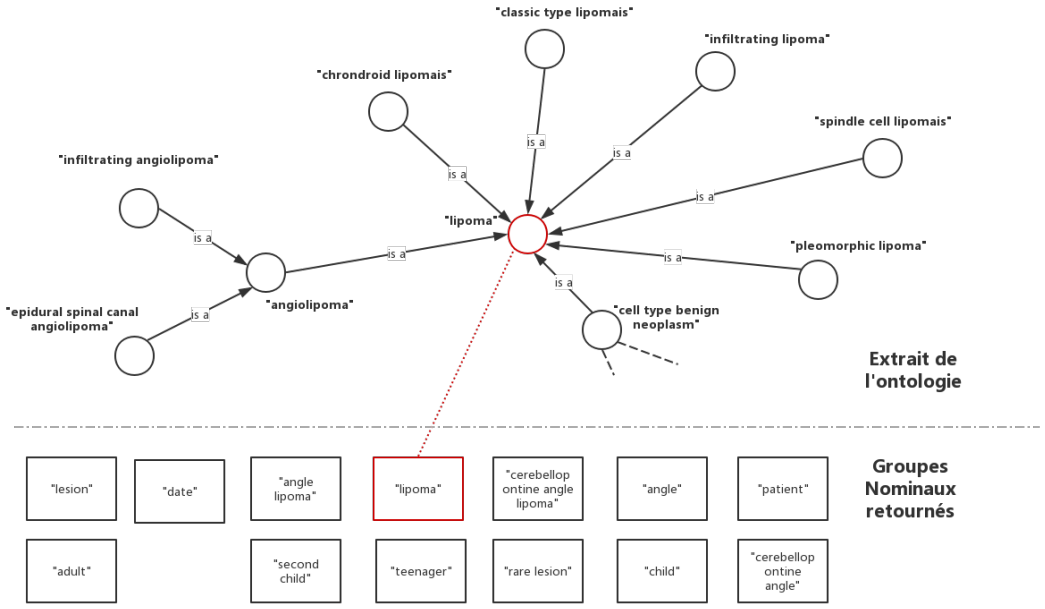


FIGURE 3.15: Recherche des concepts correspondant aux groupes nominaux contenus dans le texte 0000014.

et (Kim et al., 2006)). La section 2.2.5.6 présente dans le détail cette méthode probabiliste. Dans cette partie, je rappelle les principales composantes de cet algorithme et déroule un exemple d'apprentissage et de classification d'un nouveau texte.

3.4.1 Apprentissage

Pour rappel, l'apprentissage du classifieur naïf Bayésien consiste au calcul des estimations $\widehat{P}(t_j|C_i)$ des probabilités $P(t_j|C_i)$. L'estimation du terme t_j associée à la classe C_i est donnée par

$$\widehat{P}(t_j|C_i) = \frac{count(t_j, C_i) + 1}{\sum_{t \in V}(count(t, C_i)) + |V|} \tag{3.1}$$

Avec V l'ensemble des n concepts extraits de l'apprentissage. Le détail de la formule est donnée dans la section 2.2.5.6.

Afin d'illustrer le processus d'apprentissage, reprenons l'exemple du concept *lipoma*. Ce dernier est présent 0 fois dans les textes de la catégorie $C01$. De cette dernière a été extrait un total de 511 concepts. Enfin comme je le présenterai dans la section suivante, 1737 concepts différents ont été extraits de l'ensemble des textes d'apprentissage.

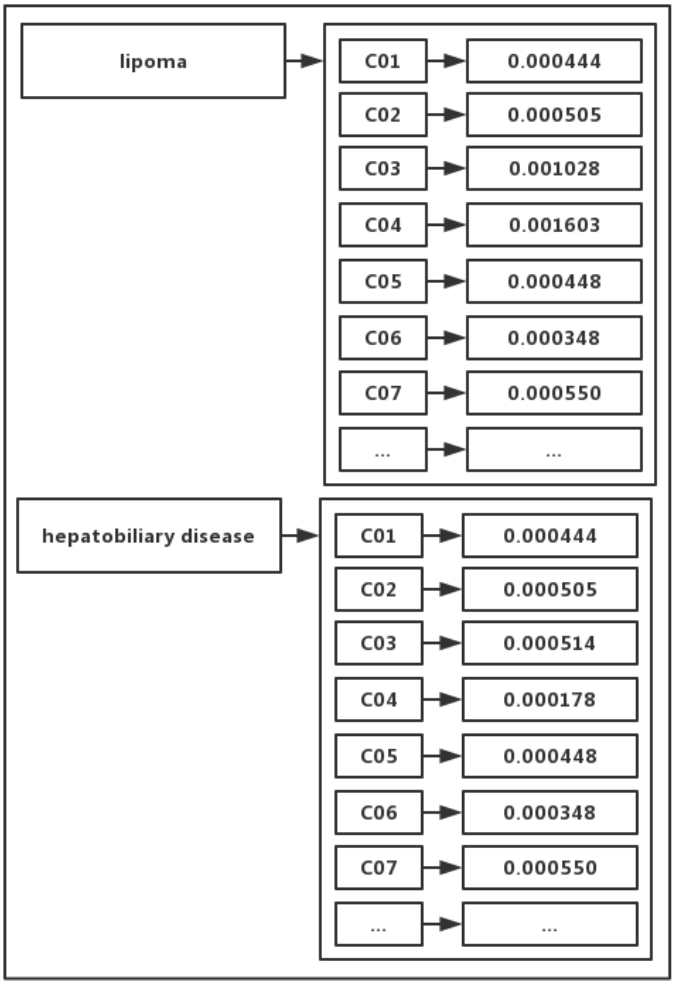


FIGURE 3.16: Extrait du modèle d'apprentissage

L'estimation de la probabilité d'apparition du concept *lipoma* dans la catégorie $C01$ est donc :

$$\widehat{P}(concept(\text{"}lipoma\text{"})|C01) = \frac{0+1}{511+1737} \approx 0.000444 \tag{3.2}$$

La probabilité estimée est d'environ 0.000444. Ce calcul est réalisé pour l'ensemble des catégories de chaque concept extrait de l'apprentissage. La figure 3.16 représente un extrait du modèle d'apprentissage.

En appliquant cette méthode d'apprentissage au corpus Ohsumed et avec la représentation des textes avec des concepts de l'ontologie Disease Ontology, le modèle obtenu est composé d'un total de 1727 concepts différents.

Comme pour toutes les autres méthodes dites inductives, le calcul du modèle d'apprentissage précède la prédiction de nouveaux textes.

3.4.2 Classification

Le classifieur bayésien est une méthode probabiliste, de ce fait l'étape de prédiction retourne les probabilités pour un nouveau texte d'appartenir à chacune des k catégories de l'analyse. La classe ayant obtenue la plus forte probabilité est sélectionnée pour labélliser le texte.

Pour rappel, le classifieur naïf Bayésien calcule la probabilité pour un texte représenté par l'ensemble de concepts Q d'appartenir à la catégorie C par :

$$P(C|Q) = P(C) \prod_{j=1}^{m} P(t_j|C) \tag{3.3}$$

Plus de précisions sont données dans la section 2.2.5.6. Les performances de cette méthode sont exposées et analysées dans les deux chapitres suivants. La classification fondée sur le modèle présenté dans ce chapitre sert de référence à mes expérimentations présentées dans les deux chapitres suivants.

3.5 Expérience

Dans le but de montrer l'intérêt de la représentation des textes avec des concepts, j'ai monté une expérience comparative. Deux modèles sont comparés, l'un utilisant la représentation en sac de mots et le second en sac de concepts. Dans une première sous partie, je présente ces deux modèles puis dans une seconde, j'expose les résultats en fonction de quatre mesures du domaine de la recherche d'informations (IR) : Le taux d'erreur, la précision, le rappel et le F1 Score.

3.5.1 Présentation des Modèles Testés

Pour cette première expérience, deux modèles sont confrontés : le premier utilise la représentation des textes par des mots et le second considère la représentation par des concepts. Ces deux modèles sont construits à l'aide du classifieur naïf Bayésien présenté dans la section 3.4 et s'appuient sur le sous ensemble du corpus de textes Ohsumed ($Ohsumed - D\&C$) présenté dans la sous-section 3.2.2 de manière à ce que l'unique différence entre ces deux modèles résident dans la méthode de représentation des textes.

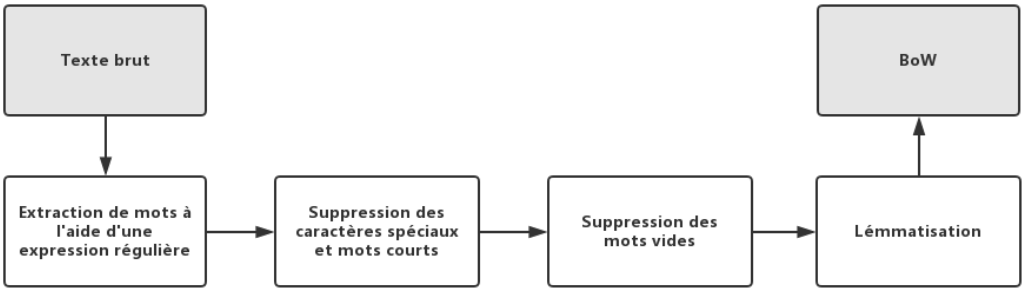


FIGURE 3.17: Liste des traitements nécessaires à la création du Bag-of-Words.

Ce dernier est divisé en deux sous-ensembles : le premier contient 70% des textes et est réservé à l'apprentissage des modèles et le second contient les 30% restants et permet le test et donc l'évaluation des modèles.

3.5.1.1 Modèle Utilisant le Paradigme BoW

La représentation des textes par des mots est largement utilisée dans la communauté. En effet, cette dernière ne nécessite aucune donnée externe et est réalisée avec des traitements simples excluant les processus lourds du TAL. La construction du BoW se déroule en trois étapes.

La première est l'extraction des mots réalisée avec une expression régulière indiquant l'espace comme séparateur de mots. La seconde est la sélection de ceux-ci. En effet, la liste obtenue lors de la première étape est nettoyée des différents caractères spéciaux (points d'interrogation, d'exclamation, de suspensions, etc.) et mots courts (généralement moins de trois lettres). Enfin, il est d'usage de supprimer certains mots du langage courant tels que les différentes formes du verbe *etre*, les déterminants, etc. Ces mots sont appelés les *motsvides* (ou *stopwords* en anglais) et ont été listé dès 1989 par (Fox, 1989). Depuis, de nombreuses listes ont été proposées dont celle de (Lewis et al., 2004)[2] listant un total de 573 mots que j'utilise pour cette expérience. La dernière étape est celle de transformation des mots et consiste à modifier les mots pour les rendre encore plus discriminants. Il est courant de lemmatiser les mots afin notamment de supprimer les différences provoquées par la conjugaison des verbes ou encore l'accord des noms. L'ensemble des traitements utilisés pour la création du BoW est présenté par la diagramme 3.17.

[2] http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

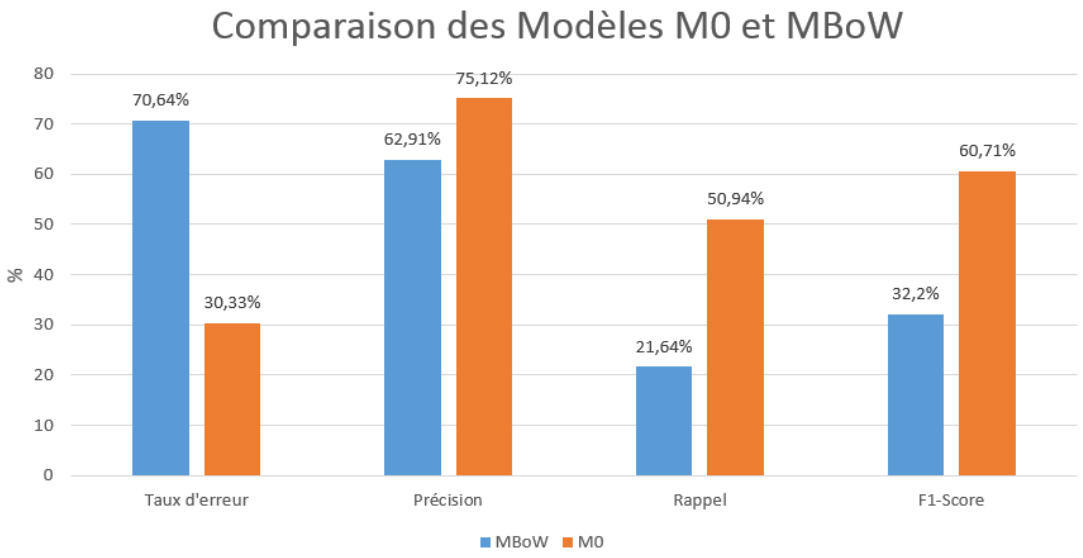


FIGURE 3.18: Comparaison des modèles M_{BoW} et M_0 à l'aide du taux d'erreur, de la précision, du rappel et du F1-Score.

Le modèle ainsi obtenu permet l'extraction de 63792 mots distincts. Pour la suite de l'expérience, je note ce modèle M_{BoW}.

3.5.1.2 Modèle Utilisant le Paradigme BoC

Comme expliqué dans les sections précédentes de ce chapitre, la représentation en BoC nécessite une ontologie de domaine. Pour la création de ce modèle, l'ontologie utilisée est Disease Ontology (DO). Plus d'informations sur ce choix est donné dans la section 3.2.1.

Représenter les textes par des concepts exige des traitements plus exigeants que ceux nécessaires à la représentation avec des mots simples. L'ensemble des processus utilisés pour la construction de ce modèle est exposé dans la section 3.3.

La création de ce modèle a permis l'extraction de 1737 concepts différents de l'ontologie DO. Ce modèle étant réutilisé dans les chapitres suivants comme étant le modèle de base, je le note M_0.

3.5.2 Résultats

La figure 3.18 présente les performances des modèles M_{BoW} et M_0 obtenues à partir du taux d'erreur, de la précision, du rappel et du F1-score.

On observe que la représentation des textes à l'aide de concepts extraits d'une ontologie de domaine permet de réduire nettement le taux d'erreur (passant de 70.64% à 30.33%), d'augmenter la précision (passant de 62.91% à 75.12%) et le rappel (passant de 21.64% à 50.94%) du modèle et accroît également la mesure du F1-Score (passant de 32.20% à 60.71%). J'en conclus que pour cette application, la représentation des textes par des concepts améliore les performances du modèle de classification.

3.6 Visualisation du Modèle

Les modèles d'apprentissage automatique sont la plupart du temps considérés comme des boites noires interprétables uniquement par la machine, seuls les résultats sont proposés à la visualisation. La simplicité de certaines méthodes autorise une représentation intuitive, c'est notamment le cas des arbres de décision (Ankerst et al., 1999). Visualiser les modèles d'apprentissage permet à l'utilisateur de comprendre la *magie* de la prédiction mais aussi et surtout au data scientist de comprendre son modèle, d'y repérer des erreurs afin de l'améliorer et de développer de nouvelles idées.

De leur côté, les ontologies sont des outils de représentation des connaissances créées pour être interprétables par les machines. Leur représentation sous forme de graphe facilite l'interprétation humaine. De nombreux logiciels permettent la visualisation de ces dernières (Katifori et al., 2007). La section 2.1.2 liste les principaux et présente notamment le plus utilisé et reconnu par la communauté : Protégé (Gennari et al., 2003). Ces logiciels se limitent à la visualisation, l'interrogation et la manipulation (création, modification, fusion, etc.) des ontologies.

Une fois la mise en place du modèle de classification présenté dans ce chapitre réalisé, j'ai souhaité le visualiser pour pouvoir l'interpréter et connaître ses limites notamment concernant la couverture des connaissances proposées par l'ontologie (je ne présente pas les résultats de la classification du modèle dans ce chapitre car cela n'a pas d'intérêt de recherche, cependant j'utilise le modèle généré pour construire la visualisation; les interprétations de la visualisation sont donc celles du modèle décrit dans ce chapitre). De ce fait, l'un de mes premiers travaux a été la construction d'une visualisation du modèle.

Dans la sous partie suivante, je présente la visualisation créée et évoque son interprétation puis dans une seconde j'expose les spécifications techniques de cette dernière.

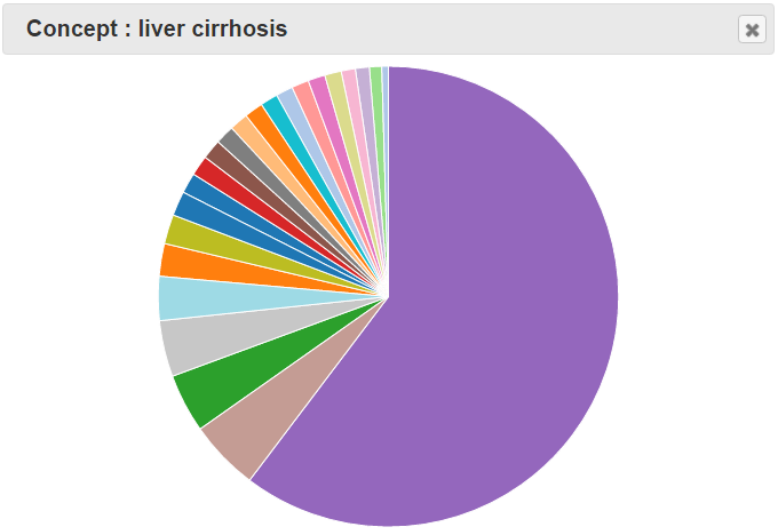


FIGURE 3.19: Visualisation des probabilités du concept *liver cirrhosis*.

3.6.1 Représentation du Modèle d'Apprentissage au Travers du Graphe de l'Ontologie

Le modèle (noté M_0 dans la section précédente) généré par le classifieur naïf Bayésien est constitué d'une liste de concepts (extraits du corpus d'apprentissage) auxquels a été associée une estimation de probabilité pour chaque catégorie de l'analyse. Pris individuellement, il est simple de représenter les probabilités associées à chaque concept. Des graphiques de base comme l'histogramme, le diagramme circulaire ou encore le diagramme de kiviat permettent cela. Le diagramme circulaire semble le meilleur choix car d'un coup d'œil, on peut observer la catégorie dominante (la part la plus grande du diagramme).

La première étape a donc été la construction d'une visualisation de concepts à l'aide de diagrammes circulaires. Le titre du diagramme correspond au label du concept permettant ainsi son identification. Les probabilités sont triées de la plus grande à la plus petite harmonisant la représentation des nombreux concepts. Au passage de la souris sur le diagramme, les noms des catégories et leur probabilité associée sont affichés. Cette interactivité permet de ne pas surcharger le graphique tout en fournissant les informations nécessaires à l'utilisateur. Pour exemple, la figure 3.19 présente le diagramme circulaire représentant le concept *liver cirrhosis*. On y observe que la catégorie violette ($C06$) est dominante. Cette visualisation permet notamment l'interprétation suivante : "Les textes contenant le concept *liver cirrhosis* ont de fortes chances de traiter du domaine de la catégorie $C06$".

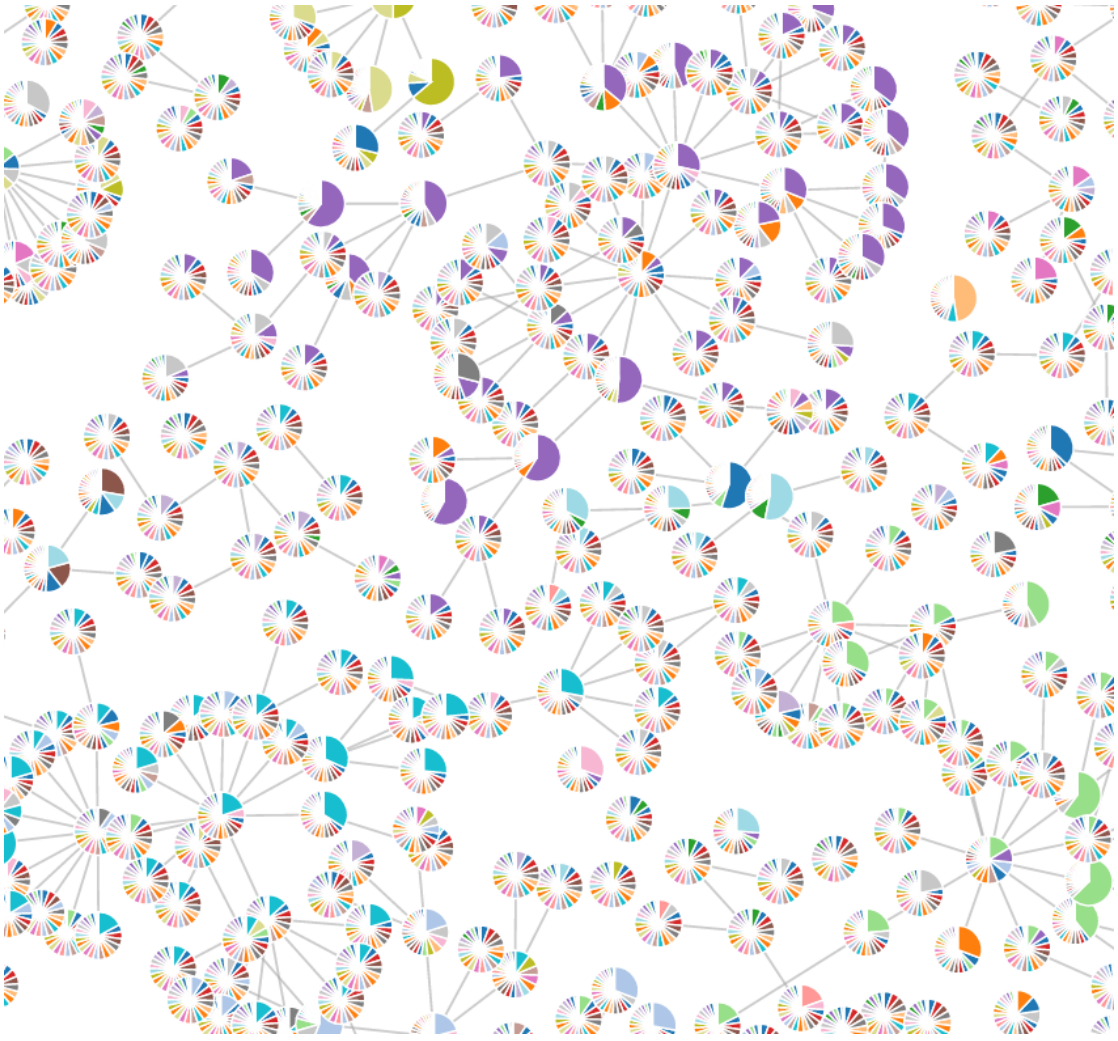


FIGURE 3.20: Visualisation d'une partie du modèle d'apprentissage.

Analyser le modèle concept par concept est une tâche laborieuse. En effet, le modèle calculé dans la section 3.4.1 en contient 1737. De plus, seule une partie des informations de l'ontologie est exposée avec cette méthode de représentation. Effectivement, les liens entre les concepts ne sont pas présents. Afin de résoudre ces deux problèmes, j'ai utilisé la représentation intuitive sous forme de graphe de l'ontologie pour organiser dans l'espace les diagrammes circulaires. La distance entre deux concepts dépend du nombre de sauts nécessaires pour passer de l'un à l'autre. Aussi, les relations sémantiques entre les concepts sont exprimées par des arcs. La figure 3.20 présente un extrait de la visualisation du modèle.

Cette visualisation permet de nouvelles interprétations. Premièrement, certains concepts semblent décrire une catégorie en particulier alors que d'autres semblent décrire de manière égale l'ensemble des catégories. J'en conclue ainsi que certains concepts sont discriminants alors que d'autres non (ou moins). Deuxièmement, certaines zones du graphe

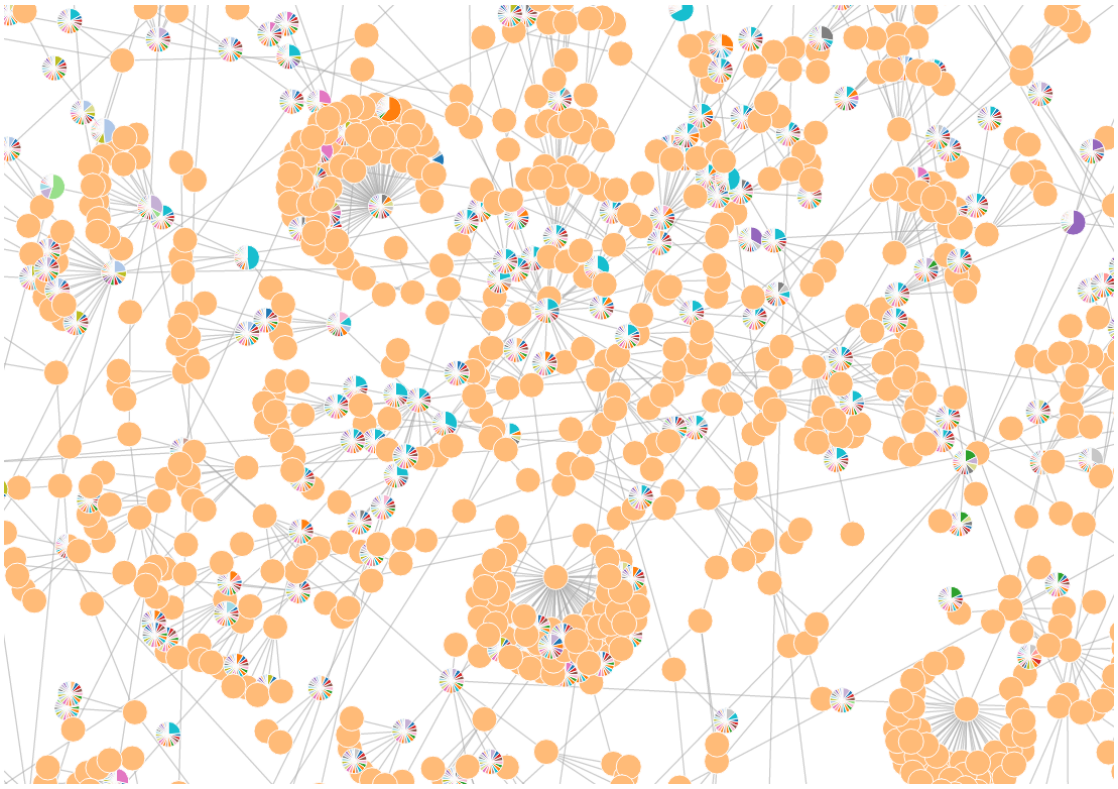


FIGURE 3.21: Visualisation d'une partie du modèle d'apprentissage avec l'ajout des concepts de l'ontologie non discriminants

semblent correspondre à des catégories. Par exemple, en bas à gauche de la figure 3.20 les concepts semblent décrire majoritairement la catégorie représentée par la couleur cyan (il s'agit de la catégorie $C08$), en haut au centre il s'agit de la catégorie violette ($C06$) et en bas à droite la catégorie vert clair ($C10$). Cette observation permet d'approuver le bien fondé du modèle d'apprentissage car l'ontologie de domaine construite par des experts expose chaque catégorie par un sous domaine de cette dernière.

Sur cette représentation, seuls les concepts extraits du corpus d'apprentissage sont affichés, expliquant le nombre de concepts "solitaires". Afin de redonner de la cohérence sémantique au graphe, j'ai pris l'initiative d'afficher les concepts non discriminants de l'analyse. La figure 3.21 présente un extrait du graphe contenant l'ensemble des concepts et relations sémantiques.

Sur cette visualisation, on observe un grand nombre de diagrammes oranges. Dans cette représentation, la couleur orange signifie que le concept correspondant n'est associé à aucune probabilité. Autrement dit, les nœuds oranges correspondent aux concepts non extraits de l'apprentissage et donc non discriminants.

Cette observation m'a mené à une conclusion : de trop nombreux concepts de l'ontologie sont inutilisés. Or, comme la figure 3.20 le montre, les catégories sont représentées par

des grappes de concepts liés entre eux par des relations sémantiques. Ainsi, j'émets l'hypothèse qu'intégrer les concepts de ces grappes dans le modèle d'apprentissage permettrait une meilleure représentation du domaine de l'ontologie au sein du modèle mais aussi améliorerait les performances des classifieurs. Dans le but de prouver ou réfuter cette hypothèse, j'ai proposé une méthode d'enrichissement du modèle d'apprentissage par l'ajout de concepts non discriminants. Le chapitre 4 est consacré à ces travaux.

3.6.2 Conception technique

Pour les raisons présentées dans la sous partie précédente, aucun logiciel de visualisation d'ontologie ne permet la création de ce genre de représentation. Cependant, de nombreux logiciels permettent la visualisation de graphes (Herman et al., 2000). Le plus reconnu et facile à prendre en main est Gephi (Bastian et al., 2009), un logiciel Français initialement créé en 2008 à l'Université de technologie de Compiègne, écrit en JAVA et qui a remporté différents prix décernés par Google et Oracle. L'intérêt principal de ce logiciel est la bibliothèque associée d'algorithmes de la théorie des graphes. Citons également les logiciels Pajek (Batagelj and Mrvar, 1998), iGraph (Csardi and Nepusz, 2006) et NetworkX (Schult and Swart, 2008).

A la fin des années 2000, la librairie jQuery (Bibeault and Kats, 2008) allait relancer le langage javascript (Flanagan, 2006) en facilitant sa manipulation. Javascript est un langage web qui s'exécute coté client et non pas serveur comme le célèbre langage PHP (Ullman, 2004) permettant l'interactivité entre l'utilisateur et l'application web sans rechargement de la page. Le succès de jQuery a mené à l'arrivée de nombreuses autres librairies écrites en javascript. Certaines d'entre elles ont pour objectif la visualisation de graphes. Citons D3.js (Zhu, 2013), Arbor.js[3] et Sigma.js[4]. Parmi ces trois librairies, D3.js, créé en 2009, est de loin la plus utilisée. Cette dernière permet la représentation et la manipulation de données numériques dynamiques en tout genre : graphes, diagrammes circulaires, histogrammes, etc. Enfin, le grand intérêt de cette librairie est la possibilité de faire interagir tous ces types de graphiques en un seul.

L'implémentation de cette visualisation a été réalisée à l'aide de la librairie D3.js pour la manipulation d'objets dynamiques, la capacité d'interaction entre différents types de graphique la documentation et la communauté des applications web. Le graphique est intégré dans une page web à l'aide des langages HTML5 (Pilgrim, 2010) et CSS3 (McFarland, 2012). Un graphe d'exemple est visible à l'adresse `http://charlieaubry.com/JC/`.

[3] `http://arborjs.org/`
[4] `http://sigmajs.org/`

Chapitre 4

Enrichissement du Modèle de Classification de Textes

4.1 Introduction

La visualisation du modèle présentée dans la section 3.6 m'a permis d'observer le modèle d'apprentissage d'un classifieur naïf Bayésien pour la classification de textes représentés par des concepts (paradigme Bag-of-Concepts). De cette observation, j'ai remarqué qu'un petit sous ensemble seulement de l'ontologie de domaine était retenu dans l'apprentissage et donc que le modèle couvrait seulement une petite partie du vocabulaire du domaine de l'étude. De cette constatation, j'émets l'hypothèse qu'augmenter la couverture des concepts de l'ontologie au sein du modèle permettrait d'améliorer les performances de celui-ci.

C'est en se basant sur ce type d'hypothèse que plusieurs recherches ont proposé des méthodes pour enrichir les caractéristiques des textes. Dans (Benkhalifa et al., 2001) les concepts d'une ontologie associée à l'étude ayant une correspondance avec une ou plusieurs des caractéristiques d'un texte sont ajoutés à sa représentation. Dans l'idée d'enrichissement d'un sac de mots avec des concepts, (Hotho et al., 2003) a proposé d'enrichir les caractéristiques des textes en les remplaçant par les concepts correspondants produisant une représentation hybride. Également, (Wang et al., 2007) a proposé une méthode pour enrichir les caractéristiques des textes avec des concepts hyperonymes (catégorie de Wikipédia[1]) non contenus dans ceux-ci. Cette dernière méthode a l'avantage d'ajouter de nouvelles connaissances externes à la représentation des textes et donc au modèle permettant d'élargir le domaine de connaissances de celui-ci. Cependant cette intégration a plusieurs limites. Premièrement, elle est réalisée au niveau de la

[1] http://wikipedia.com/

représentation des textes empêchant de s'appuyer sur les règles du modèle pour le calcul des mesures des nouveaux concepts. Deuxièmement, les caractéristiques des textes sont enrichies uniquement avec leurs hyperonymes directs. Enfin, l'utilisation des catégories de Wikipédia peut être vue comme un atout puisque ce site recouvre de nombreux domaines et s'adapte ainsi à de nombreuses applications de classification de textes, cependant une analyse spécialisée (telle que la maintenance prédictive des avions Airbus par exemple) ne serait pas supportée.

Les ontologies permettent de pallier cette dernière limite. Pour rappel, (Guarino, 1997) a présenté trois niveaux d'ontologie : ontologie globale, ontologie de domaine et ontologie d'application. Le premier niveau a pour objectif de décrire des concepts très généraux, le second vise à représenter un domaine spécifique avec des concepts ciblés et enfin le dernier niveau a pour but de décrire un champ d'application et est encore plus spécifique qu'une ontologie de domaine. Plus de détails sont donnés dans la section 2.1.1. Par définition, une ontologie générale n'est pas adaptée à la classification de textes décrivant un domaine en particulier contrairement aux ontologies de domaine et d'application.

En me basant sur les travaux d'enrichissement de (Wang et al., 2007), l'utilisation des ontologies de domaine et le modèle de classification de textes représentés par des concepts que j'ai présentés dans le chapitre 3, j'ai proposé une méthode d'enrichissement du modèle. Celle-ci est réalisée après la création du modèle dans le but d'utiliser ses règles apprises pour le calcul des mesures des nouveaux concepts.

Dans une première section, je présente le principe de la méthode énoncée ainsi que deux mesures nécessaires au calcul à la représentation des nouveaux concepts. Dans une seconde section, je présente l'expérience réalisée pour tester ma méthode (et mes deux mesures associées) à l'aide de l'ontologie DO et du corpus Ohsumed, le modèle de comparaison étant celui présenté dans le chapitre 3.

4.2 Méthode pour l'Enrichissement du Modèle

La technique proposée a pour objectif l'enrichissement du modèle exposé dans le chapitre 3. Il s'agit ainsi d'une méthode d'enrichissement de modèle d'apprentissage du classifieur naïf bayésien adaptée pour la classification de textes représentés par des concepts. Par soucis de cohérence, la notation du chapitre 3 est conservée. Ainsi, $E = \{e_1, ..., e_l\}$ désigne l'ensemble des l concepts de l'ontologie de domaine associée, $d(e_i, e_j)$ le nombre minimal de sauts nécessaires pour relier les concepts e_i et e_j et V l'ensemble des n concepts extraits de l'apprentissage. Cette nouvelle méthode introduit le paramètre d_{max} que j'expliquerai le moment venu.

d_{max}	0	1	2	3	4	5	6	7
$\|V'\|$	1737	4338	6048	6756	6972	7051	7053	7053
$\|V'\|/\|E\|$	18.78%	46.91%	65.40%	73.06%	75.39%	76.25%	76.27%	76.27%

FIGURE 4.1: Nombre de concepts du modèle après enrichissement en fonction de la distance d_{max}.

4.2.1 Principe Général

4.2.1.1 Présentation du Pseudo-Algorithme

Pour enrichir le modèle avec de nouveaux concepts, je propose d'utiliser l'ensemble des concepts non extraits de l'apprentissage (que je désigne les *concepts potentiels*) et leurs relations sémantiques les liant. Grâce à ces dernières, je suis en mesure d'évaluer la distance de chacun des concepts potentiels de leur concept du modèle le plus proche. Autrement dit pour tout e_i de E/V, on connaît la distance minimum $d(e_i, V)$. Le principe de l'enrichissement proposé est le suivant : en fonction d'une distance maximale notée d_{max}, l'ensemble des concepts potentiels distant d'au maximum d_{max} sauts de V sont ajoutés au modèle. Je note V' l'ensemble des concepts du modèle après enrichissement. A partir du modèle présenté dans le chapitre 3, le tableau 4.1 présentant le cardinal de V en fonction de d_{max} et montre l'intérêt de la méthode d'enrichissement. En effet, alors que seulement 18.78% des concepts de l'ontologie sont représentés par le modèle sans enrichissement, le taux de couverture monte à 46.91% avec une valeur $d_{max} = 1$ et 65.40% avec $d_{max} = 2$. Également, ce tableau me permet d'observer qu'à partir d'une valeur $d_{max} = 3$, le taux de concepts de l'ontologie exprimé dans le modèle se stabilise.

Fort de cette observation, je propose un algorithme se déroulant en 4 étapes majeures auquel il faut ajouter une initialisation que je représente par l'étape 0. Ce dernier est présenté par le pseudo-algorithme suivant :

- DEBUT
  - Étape 0 (Initialisation) : Initialisation des listes W et X à vide.
  - Étape 1 : Sélection d'un concept de E non inclus dans les ensembles $\{V, W, X\}$ $(E/\{V, W, X\})$. J'appelle ces concepts les *concepts potentiels*. Si aucun concept n'est retourné, on passe à l'étape 3, sinon l'algorithme passe à l'étape 2 et on note le concept sélectionné e_{new} pour la suite de l'explication.
  - Étape 2 : Si le concept sélectionné est distant d'au maximum d_{max} sauts de l'ensemble V $(d(V, e_{new}) \leq d_{max})$, alors on l'intègre au sous ensemble W. Sinon on intègre e_{new} au sous ensemble X. Retour à l'étape 2.

  - Étape 3 : Pour tous les concepts de W, calcul des estimations des probabilités d'appartenance aux catégories de l'analyse et association à ces derniers.
  - Étape 4 : Création de V' par l'ajout de l'ensemble W dans V.
- FIN

L'étape 3 de l'algorithme de la méthode d'enrichissement fait intervenir le calcul des estimations des probabilités d'appartenance aux catégories pour chaque concept non extraits de l'apprentissage mais étant jugé comme proche (en fonction de d_{max}). Les sections 4.2.2 et 4.2.3 présentent les deux mesures que j'ai proposées et que je présente pour le calcul de ces estimations.

4.2.1.2 Explication par l'Exemple

Pour accompagner ce pseudo-algorithme, je propose un exemple du déroulement des étapes 0, 1 et 2 de ce dernier avec pour paramètre $d_{max} = 1$ et un ensemble E contenant un total de 6 concepts ($\{concept1; concept2; concept3; concept4; concept5; concept6\}$) dont 2 sont inclus dans l'ensemble V ($\{concept4; concept5\}$). La figure 4.2 présente le système et le processus complet de la sélection des concepts potentiels pour l'enrichissement du modèle.

Le premier schéma présente l'initialisation du système (étape 0). On observe ainsi que les listes W et X sont vides alors que V contient les concepts 4 et 5. Ces deux concepts représentent le modèle d'apprentissage alors que les concepts 1, 2, 3 et 6 représentent les concepts potentiels de E non extraits lors de l'apprentissage. Les quatre schémas qui suivent l'initialisation correspondent aux traitements de chacun de ces 4 concepts potentiels.

Le second schéma représente l'analyse du concept 1. Ce dernier est ajouté à la liste X des concepts non retenus pour enrichir le modèle. En effet, la distance le séparant de V ($d(V, concept - 1) = 2$) est supérieur à d_{max}. Le troisième schéma correspond à l'état du système après l'analyse du concept 2. Cette fois-ci, le concept est retenu dans W car $d(V, concept - 2) \leq d_{max}$. Effectivement, ce dernier est distant de 1 saut du concept 4 inclus dans V. Le concept 3 est également retenu dans W pour la même raison que le concept 2. Enfin, le dernier concept potentiel analysé est le concept 6 et ce dernier est retenu dans W car $d(V, concept - 6) \leq d_{max}$.

Après l'analyse du concept 6, plus aucun concept de E est non inclus dans l'une des listes $\{V, W, X\}$. L'algorithme passe à l'étape 3 de l'algorithme qui consiste aux calculs des estimations de probabilités de concepts de W à l'aide de deux mesures que j'ai proposées et que je présente dans les deux sous parties suivantes.

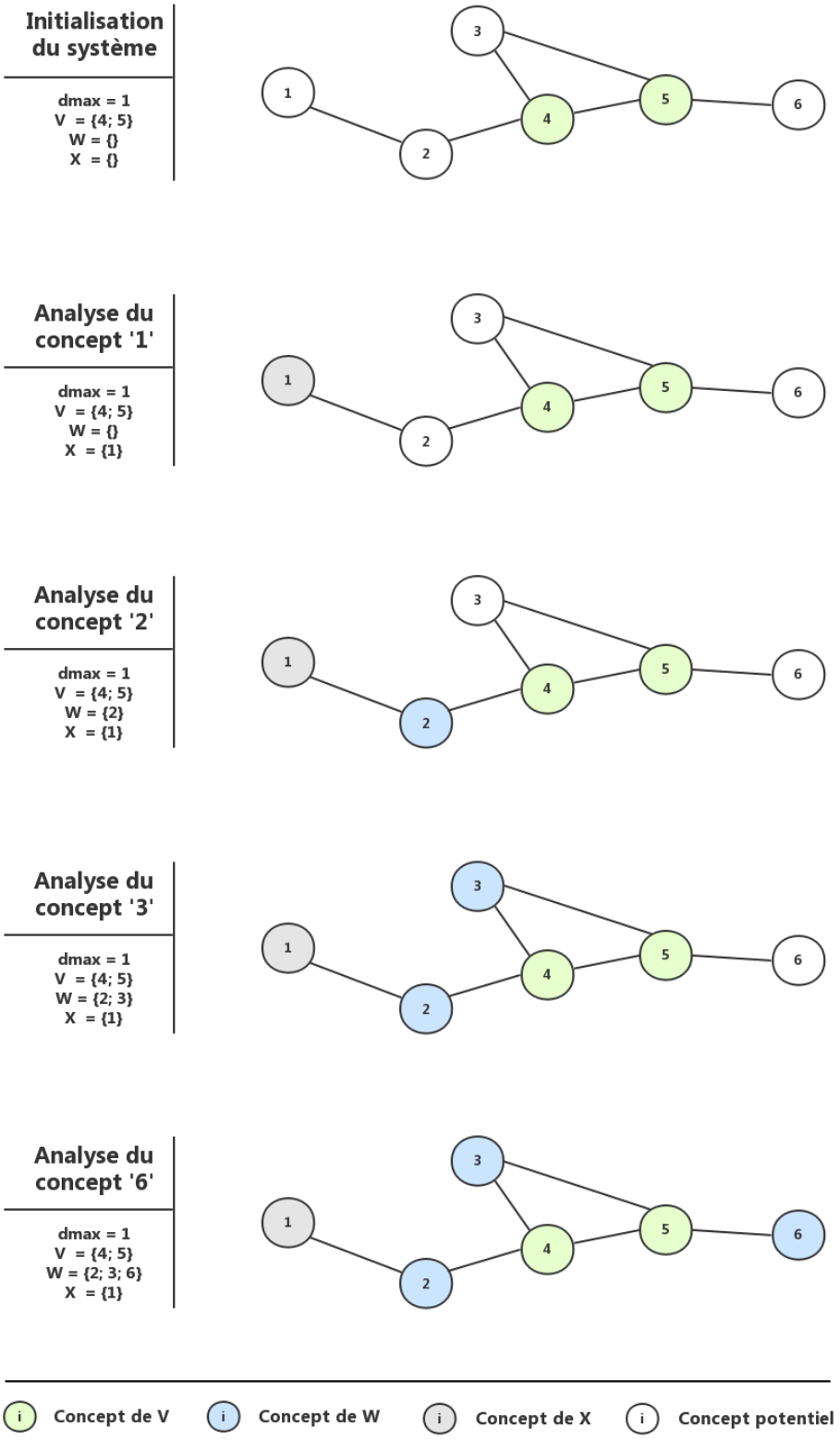


FIGURE 4.2: Exemple 1. Déroulé des étapes 0, 1 et 2 de l'algorithme.

	Concept 4	Concept 5
C_1	$\widehat{P}(concept4\|C_1) = 0.024$	$\widehat{P}(concept5\|C_1) = 0.035$
C_2	$\widehat{P}(concept4\|C_2) = 0.087$	$\widehat{P}(concept5\|C_2) = 0.012$
C_3	$\widehat{P}(concept4\|C_3) = 0.001$	$\widehat{P}(concept5\|C_3) = 0.078$
C_4	$\widehat{P}(concept4\|C_4) = 0.124$	$\widehat{P}(concept5\|C_4) = 0.009$

FIGURE 4.3: Estimation des probabilités des concepts 4 et 5 pour les catégories C_1, C_2, C_3, et C_4.

4.2.2 Mesure pour l'Enrichissement du Modèle

La mesure proposée vise à estimer les probabilités des concepts retenus lors de l'étape d'enrichissement du modèle d'apprentissage (W). Compte tenu d'un concept t_j de W, cette mesure se base sur l'ensemble des n concepts t_k de V distants d'au maximum d_{max} saut(s) de t_j. La mesure du calcul des estimations des probabilités du concept t_j d'appartenir à la classe C est donnée par :

$$\widehat{P}(t_j|C) = \frac{1}{n} * \sum_{k=1}^{n} \widehat{P}(t_k|C) \tag{4.1}$$

Pour la suite de ce document, je la note $Mesure_1$. Illustrons cette mesure en complétant l'exemple précédent présenté dans la section 4.2.1.2. Le tableau 4.3 donne la liste des estimations des probabilités des concepts 4 et 5 de V pour les catégories $C = \{C_1, C_2, C_3, C_4\}$ obtenues par la $Mesure_1$.

L'étape 3 du pseudo-algorithme exposé dans la section 4.2.1.1 a pour objectif de calculer les estimations des concepts de W à partir des estimations des probabilités des concepts de V. Par exemple, l'estimation du concept 3 inclus dans W pour la catégorie C_1 est détaillée par :

$$\begin{aligned} \widehat{P}(concept3|C_1) &= \frac{1}{2} * (\widehat{P}(concept4|C_1) + \widehat{P}(concept5|C_1)) \\ \widehat{P}(concept3|C_1) &= \frac{1}{2} * (0.024 + 0.035) = 0.0295 \end{aligned} \tag{4.2}$$

Un récapitulatif des estimations des probabilités des concepts 2, 3 et 6 à partir du tableau 4.3 et de la $Mesure_1$ est donné dans le tableau 4.4.

	Concept 2	Concept 3	Concept 6
C_1	0.024	0.0295	0.035
C_2	0.087	0.0495	0.012
C_3	0.001	0.0395	0.078
C_4	0.124	0.0665	0.009

FIGURE 4.4: Estimation des probabilités des concepts 2, 3 et 6 pour les catégories C_1, C_2, C_3, et C_4 et à l'aide de la $Mesure_1$.

4.2.3 Ajustement de la Mesure en Fonction de la Distance

Dans la sous-section précédente, j'ai présenté la première formule de calcul des estimations des probabilités pour les concepts de W. À partir de cette dernière, j'ai développé une nouvelle mesure dérivée intégrant la distance d que j'appelle $Mesure_2$. Celle-ci se base sur l'intuition suivante : plus deux concepts sont distants dans le graphe de l'ontologie, moins ils se rapportent à un sens commun. Concrètement, il s'agit d'ajouter un facteur de poids pour chacun des n concepts t_k de V dans le calcul des estimations des probabilités du concept potentiel.

Ainsi, le calcul de la probabilité du concept t_j d'appartenir à la catégorie C est donné par :

$$\widehat{P}(t_j|C) = \frac{1}{n} * \sum_{k=1}^{n} \frac{1}{d(t_k, t_j)} \widehat{P}(t_k|C) \tag{4.3}$$

Le facteur $\frac{1}{d(t_k,t_j)}$ a été ajouté à la précédente mesure ($Mesure_1$) permettant de donner plus ou moins d'importance aux concepts de V en fonction de leur distance avec le concept potentiel t_j.

Cette mesure n'a d'utilité que si d_{max} est supérieur à 1. Ainsi, avec une valeur de d_{max} égale à 2 appliquée à l'exemple de la section 4.2.1.2, l'estimation de probabilité du concept 2 d'appartenir à la classe C_1 est détaillée par :

$$\begin{aligned} \widehat{P}(concept2|C_1) &= \frac{1}{2} * (\frac{1}{d(t_k, t_j)} \widehat{P}(concept4|C_1) + \frac{1}{d(t_k, t_j)} \widehat{P}(concept5|C_1)) \\ \widehat{P}(concept2|C_1) &= \frac{1}{2} * (\frac{1}{1} * 0.024 + \frac{1}{2} * 0.035) = 0.02075 \end{aligned} \tag{4.4}$$

Un récapitulatif des estimations des probabilités des concepts 2, 3 et 6 à partir du tableau 4.3, de la $Mesure_2$ et du paramètre $d_{max} = 2$ est donné dans le tableau 4.5.

	Concept 2	Concept 3	Concept 6
C_1	0.02075	0.0295	0.0235
C_2	0.0465	0.0495	0.02775
C_3	0.02	0.0395	0.03925
C_4	0.06425	0.0665	0.0355

FIGURE 4.5: Estimation des probabilités des concepts 2, 3 et 6 pour les catégories C_1, C_2, C_3, et C_4 et à l'aide de la $Mesure_2$.

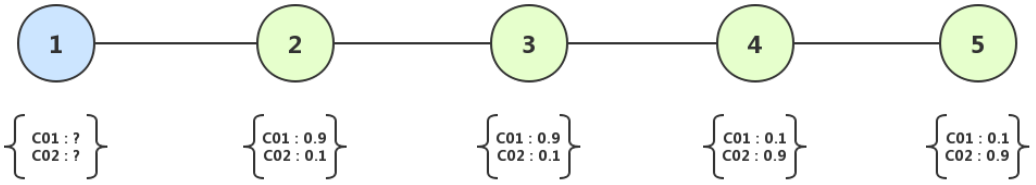


FIGURE 4.6: Exemple 2. Modèle contenant 4 concepts de V liés à un concept potentiel.

	$d_{max} = 2$		$d_{max} = 3$		$d_{max} = 4$	
	$Mesure_1$	$Mesure_2$	$Mesure_1$	$Mesure_2$	$Mesure_1$	$Mesure_2$
$C01$	0.9	0.675	0.6333	0.4611	0.5	0.3521
$C02$	0.1	0.075	0.3667	0.15	0.5	0.16875

FIGURE 4.7: Estimation des probabilités du concept 1 de l'exemple 2 en fonction de d_{max} et des mesures $Mesure_1$ et $Mesure_2$.

Afin de montrer l'intérêt de cette nouvelle mesure, je propose un nouvel exemple. La figure 4.6 montre un exemple de graphe contenant un total de 5 concepts liés sémantiquement dont 4 sont inclus dans V (les concepts 2, 3, 4 et 5) et 1 est un concept potentiel (le concept 1).

A partir de ce nouvel exemple, calculons les estimations de probabilités du concept 1 avec $d_{max} = 3$ et $d_{max} = 4$. Le tableau 4.7 résume les estimations obtenues pour les deux valeurs du paramètre d_{max} et les deux mesures proposées.

On observe que les estimations utilisant la $Mesure_2$ ont de manière générale des valeurs plus faibles que celles utilisant la première. Cependant, le calcul des probabilités d'appartenance aux différentes catégories d'un nouveau texte donné par le classifieur naïf Bayésien correspond aux produits regroupés par catégorie des estimations des concepts extraits de ce dernier. Ainsi, le niveau des estimations des probabilités des concepts du modèle n'a pas d'importance, ce sont les ratios entre les estimations des probabilités des catégories pour chaque concept qui sont discriminants.

Dans l'exemple 2, les deux concepts les plus proches (concepts 2 et 3) du concept potentiel (concept 1) définissent davantage la catégorie $C01$ ($\widehat{P}(concept2|C01) = \widehat{P}(concept3|C01) = 0.9$) que la catégorie $C02$ ($\widehat{P}(concept2|C01) = \widehat{P}(concept3|C01) = 0.1$). Cependant, les deux suivants (concepts 4 et 5) distants respectivement de 3 et 4 sauts du concept potentiel suivent le schéma inverse. Avec une valeur $d_{max} = 4$ et en utilisant la $Mesure_1$, les estimations des probabilités sont égales à 0.5 pour les classes $C01$ et $C02$ (le ratio $C01/C02 = 1$) alors qu'en utilisant la $Mesure_2$, les estimations des probabilités sont de 0.3521 pour la classe $C01$ et 0.16875 pour la classe $C02$ (le ratio $C01/C02 \simeq 2$). Autrement dit, cela signifie que la $Mesure_1$ définit le concept 1 comme étant autant décrit par la classe $C01$ que la classe $C02$ alors que la $Mesure_2$ le définit comment étant deux fois plus représenté par la classe $C01$ que la classe $C02$, ce qui est effectivement satisfaisant au regard du voisinage proche de ce dernier.

4.3 Expérience

Pour tester mes contributions sur l'enrichissement de modèles de classification de textes représentés par des concepts, j'ai construit treize modèles différents que je présente dans la sous-section suivante. Chacun de ceux-ci est évalué en fonction de quatre mesures du domaine de la recherche d'informations (IR) : Le taux d'erreur, la précision, le rappel et le F1 Score. Une seconde sous-section expose les résultats obtenus tandis que je conclue et discute ces derniers dans une troisième.

4.3.1 Présentation des Modèles Testés

Les modèles testés dépendent de deux paramètres : la technique d'enrichissement utilisée (sans enrichissement, à l'aide de la $Mesure_1$ et à l'aide de la $Mesure_2$) et la distance d_{max}. A l'aide du tableau 4.1 décrit dans la sous-section 4.2.1, j'ai identifié que l'augmentation de la valeur du paramètre d_{max} de 6 à 7 ne permettait pas l'ajout de nouveaux concepts dans le modèle (le cardinal de V' obtenu étant stabilisé à 7053). C'est pour cette raison que j'ai fait le choix de tester d_{max} pour des valeurs allant de 1 à 6. Je note M_0 le modèle de base sans enrichissement, M_1 le modèle enrichi à l'aide de la $Mesure_1$ et M_2 le modèle enrichi à l'aide de la $Mesure_2$.

Ces modèles sont construits par enrichissement du modèle présenté dans le chapitre 3, c'est à dire à l'aide du classifieur naïf Bayésien présenté dans la section 2.2.5.6, du sous-ensemble de textes labellisés extrait du corpus Ohsumed présenté en détail dans la section 3.2.2 et de l'ontologie Disease Ontology dont les spécifications ont été données dans la section 3.2.1.

Modèle	Taux d'erreur	Précision	Rappel	F1-Score
M_0	30.33	75.12	50.94	60.71
$M_1(d_{max} = 1)$	30.16	74.91	51.07	60.73
$M_1(d_{max} = 2)$	**29.77**	**75.02**	**51.57**	**61.12**
$M_1(d_{max} = 3)$	29.83	74.91	51.48	61.03
$M_1(d_{max} = 4)$	29.94	74.76	51.21	60.78
$M_1(d_{max} = 5)$	30.00	75.11	51.07	60.80
$M_1(d_{max} = 6)$	30.05	75.15	51.02	60.78
$M_2(d_{max} = 1)$	30.16	74.91	51.07	60.73
$M_2(d_{max} = 2)$	**29.77**	**75.00**	**51.60**	**61.14**
$M_2(d_{max} = 3)$	29.83	74.92	51.54	61.07
$M_2(d_{max} = 4)$	29.91	74.72	51.25	60.80
$M_2(d_{max} = 5)$	30.05	75.10	50.99	60.74
$M_2(d_{max} = 6)$	30.00	75.19	51.13	60.87

FIGURE 4.8: Évaluation des modèles M_0, M_1 et M_2.

4.3.2 Résultats

Le tableau 4.8 présente les résultats de l'expérience menée et la figure 4.9 résume ces derniers à l'aide de quatre histogrammes. Sur cette dernière, par soucis de lisibilité d_{max} est notée simplement d. Dans un premier paragraphe, je compare le modèle de base M_0 contre les modèles M_1 et M_2, puis dans un second je discute de la valeur optimale de d_{max}, dans un troisième je présente le comportement instable du score de précision et enfin, j'évalue les différences apportées par les modèles M_1 et M_2.

On observe que les modèles M_1 et M_2 réduisent le taux de mauvaise classification par rapport au modèle de base M_0 pour n'importe quelle valeur de d_{max}. Également, les modèles M_1 et M_2 dégradent le score de précision par rapport au modèle M_0 excepté pour une valeur de $d_{max} = 6$. Concernant le rappel et le F1-score, les scores obtenus par M_0 sont inférieurs à ceux obtenus par M_1 et M_2 quelle que soit la valeur de d_{max}. Ma première conclusion est que l'enrichissement des modèles permet de manière générale l'augmentation de la qualité des modèles de classification.

Que ce soit pour le taux d'erreur, le rappel et le F1-score, la valeur de d_{max} optimale est égale à 2. En effet, ces trois mesures présentent des comportements similaires : pour une valeur $d_{max} = 1$ les scores sont mauvais, le pic des meilleurs résultats est obtenu avec $d_{max} = 2$ puis à partir de $d_{max} = 3$ les scores se dégradent. Cette attitude est observée pour les modèles M_1 et M_2.

Les scores de précision se démarquent des autres. En effet, alors que le comportement remarqué sur les trois autres mesures est respecté pour des valeurs de d_{max} allant de 1



FIGURE 4.9: Histogramme des résultats généraux de l'expérience.

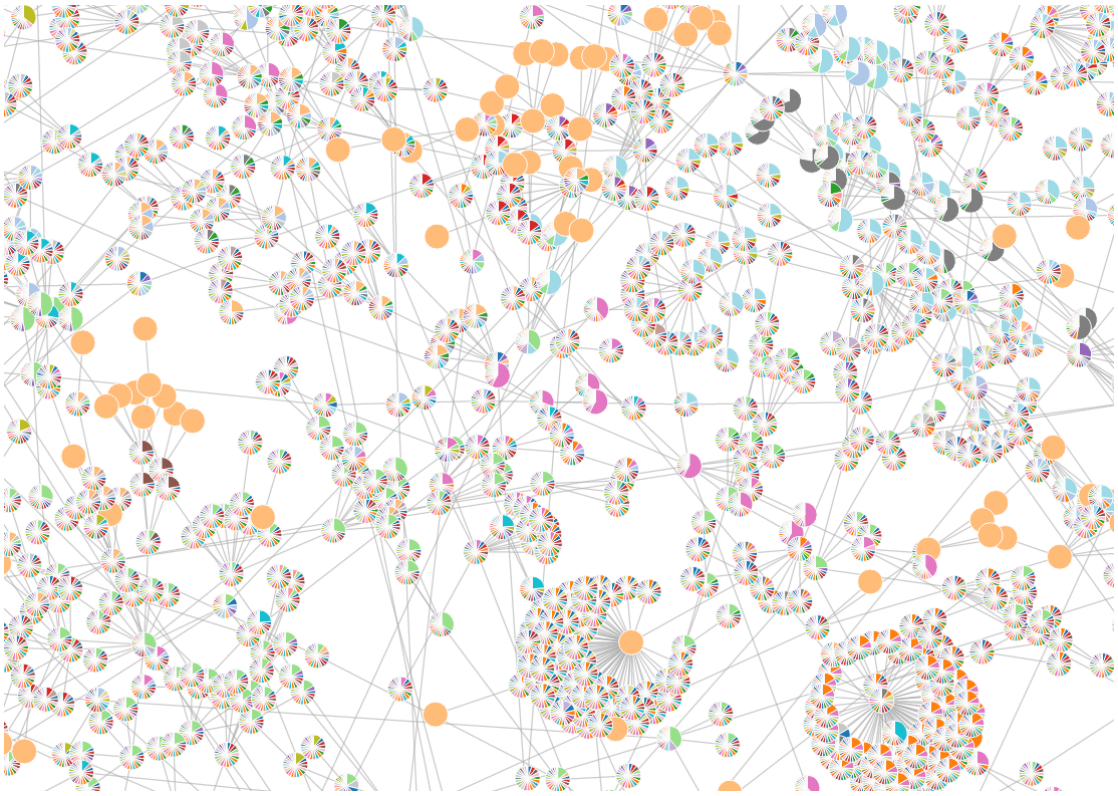


FIGURE 4.10: Extrait du graphe du modèle $M_1(d_{max} = 2)$. Les nœuds oranges sont des concepts non inclus dans le modèle.

à 4, à partir d'une valeur égale à 5, les scores de précision remontent jusqu'à dépasser celle de M_0.

La valeur de d_{max} permettant d'obtenir les meilleurs résultats, toutes mesures confondues, est $d_{max} = 2$. Compte tenu de celle-ci, d'un type de modèle ($Mesure_1$ ou $Mesure_2$) à l'autre, les résultats sont fortement similaires. Ainsi le paramètre $\frac{1}{d(t_k,t_j)}$ introduit dans la $Mesure_2$ (voir la sous-section 4.2.3) ne permet pas d'améliorer la qualité du modèle de classification comme j'en avais fait l'hypothèse. Cependant, je conclus de cette expérience que l'enrichissement du modèle par les concepts de l'ontologie permet d'améliorer les performances des modèles de classification.

La figure 4.10 présente un extrait du modèle $M_1(d_{max} = 2)$ ayant obtenu les meilleurs résultats de l'expérience. Comme l'indique le tableau 4.1, 6048 concepts sont intégrés dans ce modèle, soit 65.40% des concepts de DO. Cette visualisation permet d'observer la large couverture du modèle sur l'ontologie en comparaison à la figure 3.20 présentant la représentation du modèle de base du chapitre 3.

4.3.3 Critique des Résultats

L'expérience présentée ci-dessus a montré que mes contributions et propositions sur l'enrichissement des modèles de classification permet d'améliorer les résultats de ce dernier. Cependant, j'ai identifié quelques points critiques à amélioration concernant la méthode et l'expérience :

- L'un des partis pris de mes travaux est de représenter les textes exclusivement avec des concepts d'une ontologie de domaine. En effet, je pense que les modèles décrits par des classes sémantiques permettent une interprétation avancée de ceux-ci par la machine ouvrant la porte à de nouvelles applications telle que la visualisation interactive. C'est pour cette raison que j'ai fait le choix de ne pas comparer la méthode avec une approche autre que le sac de concepts. Cependant, il serait intéressant de proposer une nouvelle expérience en confrontant mes contributions à des méthodes utilisant la représentation en sac de mots, sac de groupes nominaux ou encore sac hybride.

- L'algorithme d'apprentissage utilisé pour construire les modèles est le classifieur naïf Bayésien. Ce dernier est largement utilisé dans la communauté au même titre que l'algorithme de la machine à vecteurs de support ou encore le réseau de neurones. Proposer une adaptation de la méthode d'enrichissement à ces derniers permettrait de juger de la généricité de l'enrichissement proposé.

- L'une des critiques principales de la représentation des textes avec des concepts d'une ontologie est la forte dépendance entre le corpus de textes et l'ontologie. Les domaines doivent être les mêmes et exprimés (à peu près) au même niveau de précision, une mauvaise association entre ces deux données d'entrée provoquerait de mauvais résultats. L'utilisation d'ontologie plus générale est une solution à ce problème, seulement ces dernières sont imprécises. Une seconde solution serait d'associer plusieurs ontologies (de domaine et d'application) afin de couvrir davantage le domaine.

- Pour l'expérience, les textes du corpus Ohsumed ont été prétraités réduisant le nombre de textes de l'analyse à 11447. Cette réduction est due à la suppression de l'ensemble des textes labellisés par au moins deux catégories et ceux ne contenant aucun concept de l'ontologie. Cependant, l'ensemble des modèles testés l'ont été à partir des mêmes jeux de données ayant subit les mêmes prétraitements.

- La recherche des nouveaux concepts étant récursive, la méthode d'enrichissement est coûteuse en temps de traitement. De plus, l'extraction des groupes nominaux à

l'aide de méthodes du traitement automatique des langues est également très coûteuse. Pour résoudre et devancer ces problèmes, j'ai proposé une implémentation distribuée utilisant les technologies du Big Data. Le chapitre 6 présente les détails de celle-ci.

- L'utilisation d'une ontologie pour la classification de textes permet d'ajouter la dimension sémantique à l'analyse. Dans mon expérience, je n'ai pas fait la différence entre les différents types de relations sémantiques liant les concepts. Le chapitre 5 se concentre sur ce sujet et une nouvelle expérience est proposée afin d'évaluer la valeur ajoutée par chacun des types de relation sémantique de l'ontologie sur le modèle.

4.4 Résumé des Contributions

Ce chapitre a présenté ma principale contribution : un algorithme d'enrichissement du modèle de classification de textes représentés par des concepts d'une ontologie de domaine ayant pour objectif d'améliorer la couverture du domaine du modèle et d'augmenter sa qualité. Cet algorithme introduit le paramètre d_{max} ainsi que deux mesures pour l'enrichissement : $Mesure_1$ et $Mesure_2$.

Grâce à une expérience, j'ai montré que l'enrichissement augmente la qualité du modèle (la qualité optimale est obtenue avec $d_{max} = 2$) et améliore nettement la couverture du domaine (stabilité obtenue à partir de $d_{max} = 3$). L'expérience a également mis en évidence que, malgré mon hypothèse, la $Mesure_2$ n'améliore pas la qualité du modèle. J'explique cela en analysant que plus d_{max} est grand, plus la $Mesure_2$ est intéressante or, la valeur optimale de d_{max} est égale à 2.

Chapitre 5

Rôle des Relations Sémantiques dans L'Enrichissement de Modèle de Classification

5.1 Introduction

L'utilisation des ontologies pour la classification de textes permet de nombreuses possibilités et applications telles que l'enrichissement sémantique et la visualisation des modèles mais pose aussi différents points d'interrogation. Le rôle des relations sémantiques dans l'enrichissement du modèle de classification est l'un de ceux-ci.

Pour rappel, une ontologie est représentée par un ensemble de concepts liés par des relations sémantiques pouvant être de différents types. Comme identifié par (Kwak and Yong, 2010), les plus communément rencontrées sont les relations d'équivalence telle que la synonymie ("chien" est synonyme de "cabot"), les relations hiérarchiques telles que l'hyperonymie ("animal" est hyperonyme de "chien"), l'hyponymie ("chien est hyponyme de "animal"), l'holonymie ("patte" est holonyme de "griffe") et la méronymie ("griffe" est méronyme de "patte"), les relations d'opposition telle que l'antonymie ("beau" est antonyme de "moche") ou encore les relations d'association ("banque" est associé à "argent" sans qu'il n'y ait de relation de hiérarchie, d'équivalence ou d'opposition entre ces deux termes).

Dans (Zheng et al., 2009), Zheng propose une étude sur l'implication des relations sémantiques pour l'enrichissement des modèles de clustering. Dans cette dernière, les relations d'hyperonymie, d'hyponymie, d'holonymie et de méronymie sont étudiées, les résultats

montrent que c'est principalement les relations d'hyperonymie qui améliorent la qualité des modèles suivi des relations d'hyponymie puis de méronymie et enfin d'holonymie.

Mes contributions sur l'enrichissement de modèle de classification supervisée de textes représentés par des concepts présentées dans le chapitre 4 m'ont logiquement conduit à m'intéresser à l'importance des relations sémantiques pour l'enrichissement des modèles en fonction de leur type.

Dans ce chapitre, je présente l'adaptation de la méthode proposée dans le chapitre précédent pour l'analyse par type de relation dans une première section puis j'expose l'expérience dans une seconde.

5.2 Analyse par Type de Relation Sémantique

5.2.1 Contexte des Recherches

L'analyse du rôle des relations sémantiques dans l'enrichissement est la suite logique des travaux présentés dans le chapitre 4, permettant d'approfondir et d'évaluer les spécificités des ontologies sur la classification de textes. Dans le but de conserver une cohérence avec les deux précédents chapitres, cette nouvelle analyse est supportée par le modèle présenté dans le chapitre 3, la méthode d'enrichissement utilisée est décrite dans la partie 4.2.1.

Les résultats de l'expérience menée dans le chapitre précédent ont montré que les $Mesure_1$ et $Mesure_2$ testées pour enrichir les modèles d'apprentissage, permettent d'obtenir de meilleurs résultats par rapport au modèle de base M_0 (sans enrichissement). Cependant aucune différence importante n'a été relevée entre les modèles enrichis par l'une ou l'autre mesure, de plus, la $Mesure_1$ étant la plus simple et la plus générique (pour rappel la $Mesure_2$ est une extension de $Mesure_1$), j'ai fait le choix de l'utiliser pour ces nouvelles recherches. Pour rappel, la $Mesure_1$ est donnée par :

$$\widehat{P}(t_j|C) = \frac{1}{n} * \sum_{k=1}^{n} \widehat{P}(t_k|C) \tag{5.1}$$

Le modèle d'apprentissage ainsi utilisé est construit grâce au classifieur naïf Bayésien et le paradigme de représentation des textes sélectionné est le sac de concepts. Le corpus de textes Ohsumed et l'ontologie de domaine Disease Ontology sont également conservés. L'ensemble des notations vues dans les chapitres 3 et 4 est conservé dans celui-ci.

```
<!-- http://purl.obolibrary.org/obo/DOID_0050242 -->

<owl:Class rdf:about="http://purl.obolibrary.org/obo/DOID_0050242">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        primary amebic meningoencephalitis
    </rdfs:label>
    <rdfs:subClassOf rdf:resource="http://purl.obolibrary.org/obo/DOID_2789"/>
    <obo:IAO_0000115 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        A parasitic protozoa infectious disease that involves infection
        of the central nervous system caused by Naegleria fowleri.
        The symptoms include headache, nausea,
        rigidity of the neck muscles, vomiting, delirium, seizures and coma.
    </obo:IAO_0000115>
    <oboInOwl:id rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        DOID:0050242
    </oboInOwl:id>
    <oboInOwl:hasExactSynonym rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        Naegleria fowleri infection
    </oboInOwl:hasExactSynonym>
    <oboInOwl:hasOBONamespace rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        disease_ontology
    </oboInOwl:hasOBONamespace>
</owl:Class>
```

FIGURE 5.1: Description de la classe du concept *DOID_0050242*.

5.2.2 Choix et Analyse des Types de Relations Sémantiques

A l'aide de sa structure OWL, l'ontologie DO représente trois types de relations sémantiques : la synonymie, l'hyperonymie et l'hyponymie. Les liens de synonymie sont décrits par la balise *hasExactSynonym* identifiée par l'espace de nom *xmlns:obo="http://purl.obolibrary.org/obo/"*. Aussi, grâce à la balise *subClassOf* de l'espace de nom *xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"*, une classe sémantique est liée par relation hiérarchique à une autre.

Par exemple, le concept dont l'identifiant est *DOID_0050242* a pour synonyme *"Naegleria fowleri infection"* et pour hyperonyme le concept *DOID_2789*. La figure 5.1 présente la description de la classe du concept *DOID_0050242* et permet de visualiser cet exemple.

La synonymie n'étant pas exprimée de manière à lier des concepts entre eux, seules les relations hiérarchiques d'hyperonymie et d'hyponymie sont de nature à enrichir le modèle. C'est pour cette raison que j'ai sélectionné ces deux types de relations dans mon analyse.

Comme exposé dans le chapitre précédent, la profondeur du voisinage est le paramètre déterminant de l'enrichissement des modèles. Plus cette profondeur d_{max} est grande, plus le nombre de concepts représentant le modèle augmente, permettant ainsi de mieux couvrir le domaine de l'analyse. Le tableau 5.2 présente le cardinal de V (pour rappel, il s'agit du nombre de concepts représentés dans le modèle d'apprentissage) en fonction

d_{max}	0	1	2	3	4	5	6
+ hyperonymes	1737	4147	5388	6113	6607	6906	7003
+ hyponymes	1737	2087	2122	2125	2126	2126	2126
+ hyperonymes et hyponymes	1737	4338	6048	6756	6972	7051	7053

FIGURE 5.2: Cardinal de V en fonction des types de relation sémantique.

des types des relations sémantiques utilisés pour l'enrichissement. On observe ainsi que le nombre de concepts des modèles enrichis par les hyponymes est borné à partir d'une valeur $d_{max} = 4$ alors que celui des modèles enrichis par les hyperonymes ne l'est pas.

De cette courte analyse préliminaire, j'en conclue que les concepts extraits de l'apprentissage sont suffisamment spécifiques pour se retrouver au bas de la hiérarchie de l'ontologie DO. Cela conforte le choix pour ce genre d'analyse des ontologies de domaine qui décrivent précisément les termes d'un domaine contre les ontologies générales.

5.2.3 Adaptation de l'Algorithme d'Enrichissement

L'algorithme d'enrichissement du modèle par les concepts voisins présenté dans la section 4.2.1 contient quatre étapes à laquelle il est nécessaire d'ajouter une étape d'initialisation (appelé étape 0). La première consiste à la sélection d'un concept dit potentiel (non inclus dans V), la seconde définit si le concept est conservé pour être ajouté au modèle ou rejeté (inclus dans W ou dans X respectivement), la troisième calcule les estimations des probabilités des nouveaux concepts et la dernière ajoute les nouveaux concepts au modèle (inclusion de W dans V).

La prise en considération du type des relations sémantiques est un nouveau paramètre que je note r et qui s'intègre à l'étape 2. Comme expliqué dans la sous partie précédente, DO propose deux valeurs de r, soit $r = \{hyperonyme, hyponyme\}$. Alors que la méthode de base considère l'ensemble des liens (et donc des type de relations sémantiques) pour la recherche des nouveaux concepts, cette nouvelle méthode filtre ces derniers en fonction de r.

La figure 5.3 présente un exemple de modèle contenant quatre concepts dans l'ensemble V associés à leurs estimations des probabilités des deux catégories $C01$ et $C02$ et un concept potentiel.

A partir de cette figure, on observe que le concept 1 est hyperonyme du concept 2, le concept 2 est hyperonyme des concepts 3 et 4, le concept 4 est hyperonyme du concept 5 mais aussi que le concept 5 est hyponyme du concept 4, les concepts 3 et 4 sont hyponymes du concept 2 et enfin que le concept 2 est hyponyme du concept 1.

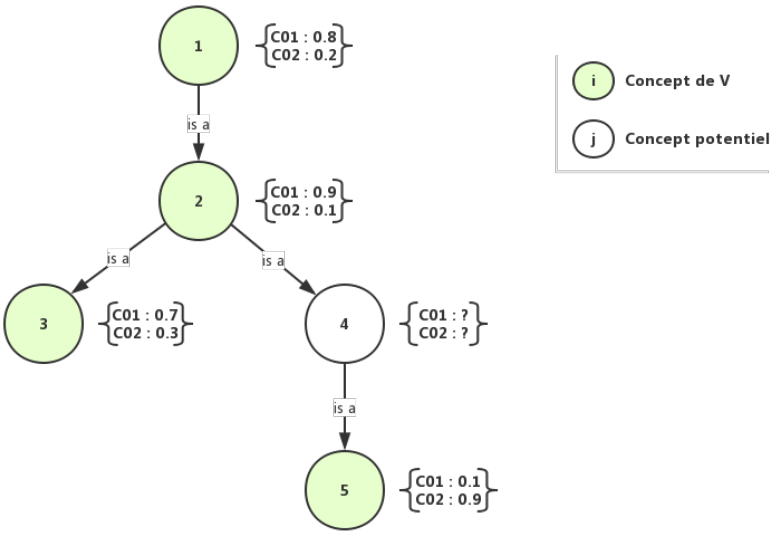


FIGURE 5.3: Exemple de l'étape d'enrichissement du modèle en fonction du type de relation sémantique.

Par exemple, considérons le calcul des estimations des probabilités du concept 4 avec une valeur $d_{max} = 2$. Suivant le choix du type de relation sélectionné r, les estimations sont données par :

- Avec $r = \{hyperonyme\}$.
  - $\widehat{P}(concept4|C01) = \frac{1}{2} * (\widehat{P}(concept2|C01) + \widehat{P}(concept1|C01)) = 0.85$
  - $\widehat{P}(concept4|C02) = \frac{1}{2} * (\widehat{P}(concept2|C02) + \widehat{P}(concept1|C02)) = 0.15$
- Avec $r = \{hyponyme\}$.
  - $\widehat{P}(concept4|C01) = \frac{1}{1} * (\widehat{P}(concept5|C01) = 0.1$
  - $\widehat{P}(concept4|C02) = \frac{1}{1} * (\widehat{P}(concept5|C02) = 0.9$

On observe ainsi que les estimations de probabilités du concept 3 ne sont pas utilisées pour le calcul des estimations du concept 4 pour les deux valeurs possibles de r.

5.3 Expérience

Cette expérience a pour objectif d'évaluer les apports des relations d'hyperonymie d'une part et d'hyponymie d'autre part dans le contexte de l'enrichissement du modèle présenté dans le chapitre 4. Dans cette optique, 19 modèles ont été évalués à l'aide de quatre

mesures du domaine de la recherche d'informations (IR) : le taux d'erreur, la précision, le rappel et le F1 Score. Dans la première partie de cette expérience, je présente les modèles mis en compétition, puis dans la seconde, j'expose les résultats et apporte quelques interprétations et je critique ces derniers dans une troisième.

5.3.1 Présentation des Modèles Testés

Tout comme l'expérience du chapitre précédent, les modèles sont évalués en fonction du paramètre d_{max} avec des valeurs allant de 1 à 6 mais également du paramètre r. Trois valeurs de r sont proposées : le modèle M_3 prend en paramètre $r = \{hyperonyme\}$ et le modèle M_4 a une valeur $r = \{hyponyme\}$. Le modèle M_1 correspond au modèle du même nom vu au chapitre précédent et, ne prenant pas en compte le type des relations sémantiques, il est équivalent d'affirmer qu'il prend en paramètre $r = \{hyperonyme, hyponyme\}$. Enfin, le modèle de base M_0 correspond également au modèle du même nom utilisé dans l'expérience précédente et sert de point de comparaison.

5.3.2 Résultats

Le tableau 5.4 expose les résultats de l'expérience et la figure 5.5 les résume en regroupant quatre histogrammes (un par mesure). Par soucis de cohérence, les résultats sont présentés de la même manière que ceux de l'expérience précédente.

Tout d'abord, on observe que le modèle M_4 obtient des résultats très réguliers pour toutes les valeurs de d_{max}. Cela est cohérent avec les informations apportées par le tableau 5.2 qui présente le cardinal de V en fonction de d_{max}. Pour rappel, on y avait observé que V était borné pour ce modèle. On observe aussi que pour les quatre mesures relevées, les scores du modèle M_4 se rapproche fortement de ceux du modèle de base M_0. J'interprète que l'enrichissement par les hyponymes ne modifie que très peu le modèle de base.

Le modèle M_3 permet d'obtenir de bons résultats, comparables à ceux de M_1, et donc de meilleure qualité que M_0. Cette observation est correcte pour des valeurs de d_{max} comprises entre 1 et 4. En effet, passée cette limite, l'ensemble des scores se dégradent fortement excepté pour la précision qui fait office de mesure outsider une fois de plus. Je note que l'ensemble des scores les plus mauvais sont obtenus par ce modèle, j'en déduis l'importance du paramètre d_{max} avec ce dernier. Mon interprétation de cette observation est la suivante : les concepts hauts dans la hiérarchie de l'ontologie sont suffisamment généraux pour ne pas discriminer les textes les contenant et ainsi dégrader les résultats.

En conclusion, en fonction de d_{max}, les trois modèles M_1, M_3 et M_4 ont des comportements différents : M_1 est variable pour des valeurs faibles de d_{max} puis se stabilise,

Modèle	Taux d'erreur	Précision	Rappel	F1-Score
M_0	30.33	75.12	50.94	60.71
$M_1(d_{max} = 1)$	30.16	74.91	51.07	60.73
$M_1(d_{max} = 2)$	**29.77**	**75.02**	**51.57**	**61.12**
$M_1(d_{max} = 3)$	29.83	74.91	51.48	61.03
$M_1(d_{max} = 4)$	29.94	74.76	51.21	60.78
$M_1(d_{max} = 5)$	30.00	75.11	51.07	60.80
$M_1(d_{max} = 6)$	30.05	75.15	51.02	60.78
$M_3(d_{max} = 1)$	30.16	74.91	51.16	60.80
$M_3(d_{max} = 2)$	**29.83**	**74.68**	**51.61**	**61.04**
$M_3(d_{max} = 3)$	29.88	74.87	51.46	60.99
$M_3(d_{max} = 4)$	30.11	74.82	51.05	60.69
$M_3(d_{max} = 5)$	30.33	74.96	50.65	60.46
$M_3(d_{max} = 6)$	30.44	74.97	50.55	60.39
$M_4(d_{max} = 1)$	30.30	75.10	50.95	60.71
$M_4(d_{max} = 2)$	30.30	75.12	50.95	60.72
$M_4(d_{max} = 3)$	30.30	75.12	50.95	60.72
$M_4(d_{max} = 4)$	30.30	75.12	50.95	60.72
$M_4(d_{max} = 5)$	30.30	75.12	50.95	60.72
$M_4(d_{max} = 6)$	30.30	75.12	50.95	60.72

FIGURE 5.4: Evaluation des modèles M_0, M_1, M_3 et M_4.

M_3 est variable pour des valeurs de d_{max} faibles (suivant le même schéma que M_1 à ce niveau) puis les scores se dégradent rapidement et enfin M_4 est le modèle le plus stable. Si la stabilité de M_4 et le comportement de M_3 sont faciles à interpréter, celui de M_1 est plus compliqué. Concernant ce dernier, mon explication est la suivante : la stabilité apparente pour des valeurs suffisamment haute de d_{max} est due à la prise en compte des hyponymes des hyperonymes. En effet, M_1 prend en compte les hyperonymes et les hyponymes, suivant la valeur de d_{max}, M_1 peut être enrichi par les concepts "frères" de V. Ces derniers, semblant apporter de bons résultats, permettent de masquer les mauvaises estimations des probabilités des hyperonymes de haut niveaux. Pour reprendre l'exemple de la figure 5.3, le modèle M_1 a l'avantage sur le modèle M_3 d'avoir intégré les estimations des probabilités du concept 3 dans le calcul de celles du concept 4.

La figure 5.6 présente les graphes des modèles $M_4(d_{max} = 2)$ et $M_3(d_{max} = 2)$. Celle-ci permet d'observer la grande différence de couverture du modèle en fonction du type de relation sémantique utilisé pour l'enrichissement.

5.3.3 Critique des Résultats

Cette contribution a permis de montrer le rôle des relations sémantiques dans l'enrichissement des modèles de classification de textes représentés par des concepts.

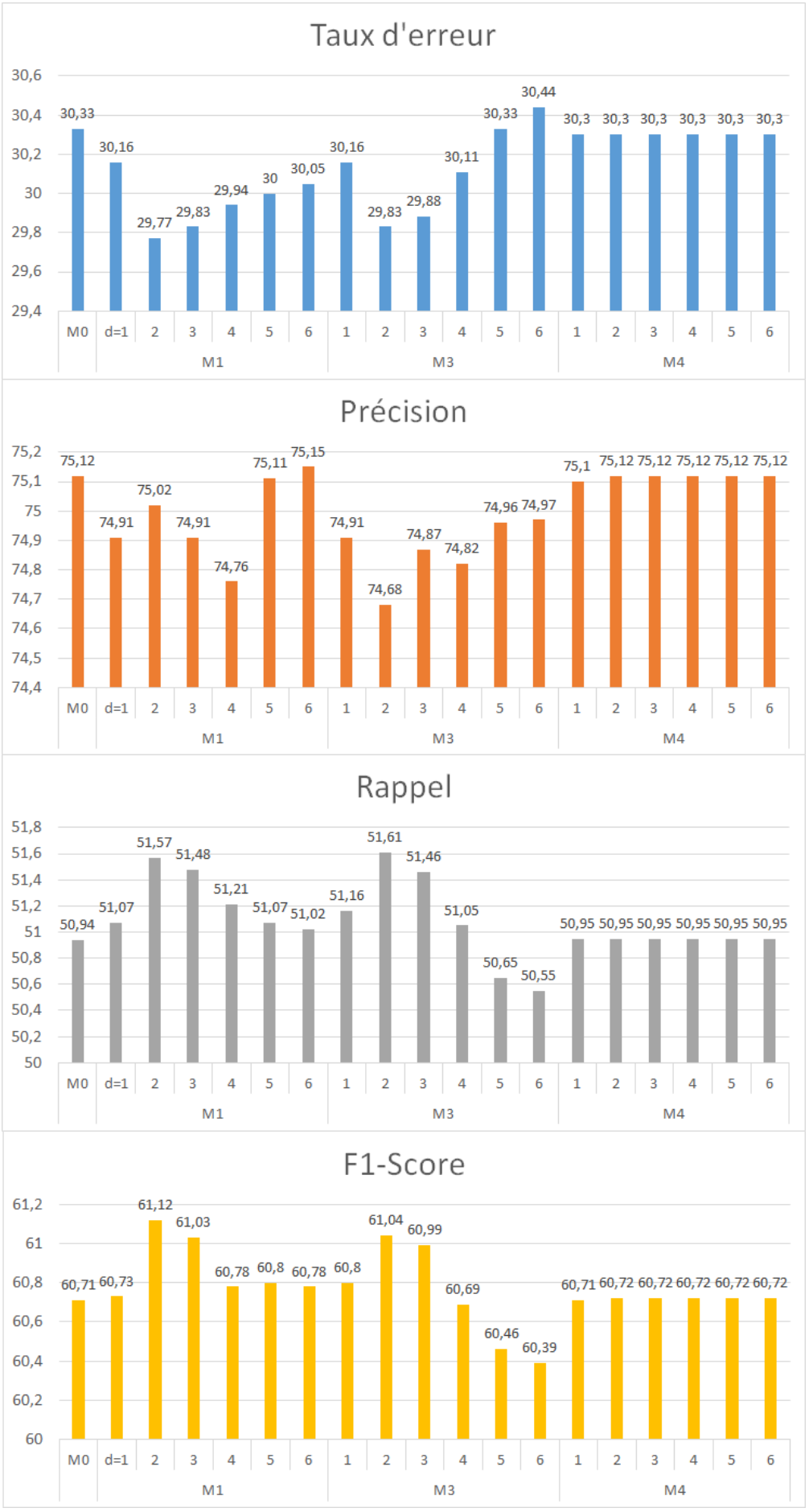


FIGURE 5.5: Exemple de l'étape d'enrichissement du modèle en fonction du type de relation sémantique.

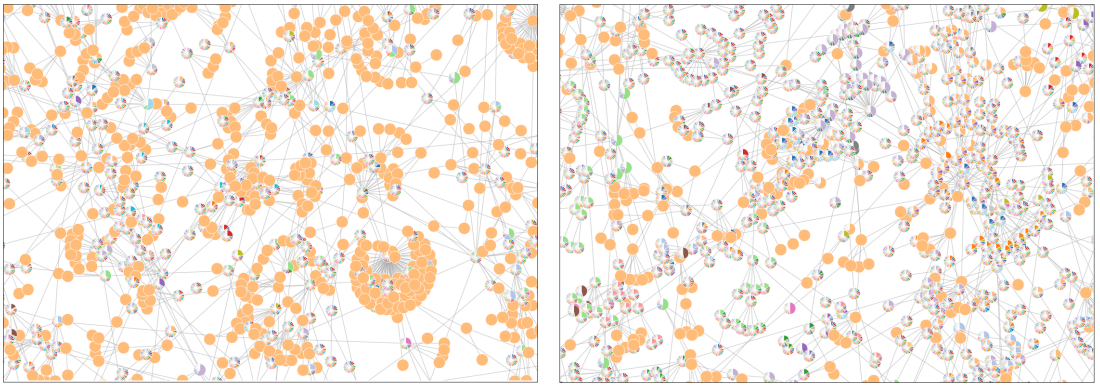


FIGURE 5.6: Extrait des graphes des modèle $M_4(d_{max} = 2)$ (à gauche) et $M_3(d_{max} = 2)$ (à droite). Les nœuds oranges sont des concepts non inclus dans les modèles.

Cependant, j'ai identifié plusieurs limites que je discute ci-après :

- La première limite, et sans doute l'une des plus importantes, est le nombre de types de relation sémantiques évalué puisque seulement les relations hiérarchiques d'hyperonymie et d'hyponymie ont pu être testées. Cette limite est due à ma volonté de proposer une suite aux contributions présentées dans le chapitre 4. Pour ces dernières, j'ai fondé mes recherches sur l'ontologie DO qui ne contient que des relations hiérarchiques entre ses concepts.
- La conclusion de l'expérience est que les concepts frères (hyponyme d'hyperonyme) semblent contribuer à l'amélioration des scores des modèles. Il serait intéressant de proposer une nouvelle expérience permettant de réfuter ou confirmer cette hypothèse.

5.4 Résumé des Contributions

Ce chapitre présente ma deuxième contribution. L'objectif de celle-ci est l'analyse de l'influence du type des relations sémantiques dans l'enrichissement du modèle présenté dans le précédent chapitre. Cette analyse introduit un nouveau paramètre r décrivant le type de la relation sémantique sélectionnée pour l'enrichissement.

Grâce à une expérience, j'ai montré l'intérêt de chacune des relations sémantiques analysées (l'hyperonymie et l'hyponymie). Ainsi, pour chacune d'elle, l'enrichissement permet d'augmenter la qualité du modèle de classification et la couverture du domaine. Cependant, ces améliorations sont plus faibles que lors de l'enrichissement intégrant les deux types de relations. Ma conclusion est que les concepts frères (hyperonyme d'hyponyme et hyponyme d'hyperonyme) permettent d'améliorer les modèles.

Chapitre 6

Amélioration des Performances et Conception Technique

Les performances des méthodes de classification est un facteur de plus en plus important avec l'accroissement exponentielle des données dans les bases. En effet, d'une application à l'autre, le nombre de données peut exploser impliquant une conception adaptée capable de passer à l'échelle. Pour résoudre ce problème, il est usuel de paralléliser les calculs les plus lourds de l'application sur les différents processeurs (threads, cœurs) de la machine sur laquelle s'exécute cette dernière. Cependant cette méthode est limitée par la capacité de la machine. La solution à ce nouveau problème a été initiée par l'entreprise Google vers la fin des années 2000. Il ne s'agit plus de paralléliser les calculs sur les différents cœurs de la machine mais de les distribuer sur un ensemble de machines en réseau (cluster). Paradoxalement, l'amélioration des performances ne s'étudie plus une fois l'application terminée mais dès le début de sa conception.

La distribution des calculs a l'inconvénient d'être complexe à mettre en place et demande un travail spécifique d'architecture des systèmes mais a le grand avantage de permettre à n'importe quelle application de passer à l'échelle (scalabilité). Ainsi, une application de classification de textes développée pour ce genre d'architecture peut traiter 10 ou 10 milliards de textes sans modifier le code source, seule l'infrastructure informatique est à adapter (peu de textes = peu de machines dans le réseau; beaucoup de textes = ajout de machines au réseau).

Suite à la publication des articles fondateurs de cette nouvelle technique de calculs distribués (Dean and Ghemawat, 2008) et (Chang et al., 2008), la communauté a mis en place différents frameworks tels que Hadoop (White, 2012) et Spark (Spark, 2014). Ces derniers ont pour objectif de rendre le processus de distribution des calculs transparent

au développeur, facilitant ainsi son utilisation. Les méthodes d'apprentissage automatique étant demandeuses de toujours plus de données pour mieux généraliser les modèles d'apprentissage, elles ont été parmi les premières à disposer de librairies et d'outils se basant sur ces frameworks : Mahout (Owen et al., 2012) pour Apache Hadoop et (Meng et al., 2016) pour Apache Spark.

Ces librairies utilisent des types de données spécifiques tel que les *Resilient Distributed Datasets* (Zaharia et al., 2012) dans le cas de Spark. Le développeur souhaitant distribuer les calculs sur ses données a la contrainte de transformer celles-ci au format RDD. Ces types de données deviennent ainsi la base du programme et oblige le développeur à penser son programme pour l'intégration de ces frameworks dès la création de l'application.

Dans ce chapitre, je commence dans une première section par la présentation d'une analyse des performances en temps de calcul de la méthode présentée dans le chapitre 3 et expose la conception technique fondée sur l'utilisation du framework Spark dans une seconde section.

6.1 Amélioration des Performances

Lors de la première construction de la méthode exposée dans le chapitre 3, j'ai observé un temps d'exécution très long au niveau de la classification des nouveaux textes et de l'apprentissage. Suite à cette constatation, j'ai entrepris une analyse afin de comprendre la source du problème.

Dans une première sous partie je présente une courte analyse m'ayant permis d'identifier l'origine de la lenteur de l'application, puis dans une seconde je propose l'utilisation d'un framework de distribution me permettant de résoudre ce problème.

6.1.1 Identification Des Temps d'Exécution

L'algorithme d'apprentissage automatique utilisé pour l'application de classification de textes dans le chapitre 3 est le classifieur naïf Bayésien. Ce dernier appartient à la catégorie des méthodes inductives nécessitant l'étape préliminaire de création du modèle d'apprentissage précédent l'opération de classification de nouveaux textes. J'ai observé la lenteur de l'application autant sur la partie apprentissage que classification, l'étape la plus simple étant celle de classification puisque celle-ci ne se divise qu'en deux sous-étapes : l'extraction des caractéristiques suivie du calcul des estimations des probabilités à l'aide du modèle.

Nb de lignes	TE en mode distribué	TE en mode séquentielle
10	17 sec	15 sec
25	29 sec	40 sec
50	42 sec	59 sec
100	79 sec	112 sec
200	139 sec	217 sec
400	241 sec	452 sec
800	439 sec	861 sec
1600	1017 sec	1801 sec
3200	2078 sec	3318 sec

FIGURE 6.1: Temps d'exécution (TE) par mode de calcul.

L'analyse de la durée d'exécution de ces deux sous-étapes a donné les résultats suivants : pour un total de 217 textes analysés, l'extraction des caractéristiques a duré 387002 millisecondes soit une moyenne de 1783 millisecondes par texte alors que les calculs des estimations de probabilités ont duré un total de 12 millisecondes soit une moyenne de 55299 nanosecondes par texte. Ainsi, l'extraction des caractéristiques dure plus de 30000 fois plus longtemps que l'estimation des probabilités.

Pour rappel, la méthode d'extraction des caractéristiques est réalisée à l'aide de techniques de traitement automatique du langage et est décrite en détail dans la section 3.3. Cette observation rejoint l'analyse de (Agerri et al., 2015), en effet, les techniques du TAL sont généralement coûteuses en temps de traitement et en occupation de la mémoire. La sous partie suivante présente la méthode utilisée pour résoudre ce problème.

6.1.2 Distribution des Traitements

La sous partie précédente a montré que le goulot d'étranglement du programme de classification de textes provient de l'extraction des caractéristiques. Cette phase contient cinq étapes séquentielles, prend en entrée un texte brut et retourne une liste de groupes nominaux en sortie. La détection des groupes nominaux est réalisée phrase par phrase de façon indépendante, c'est d'ailleurs pour cette raison que la première étape du processus est la détection des phrases avec des expressions régulières.

Le traitement des phrases étant indépendant, cela laisse la possibilité de distribuer ces phrases sur les différents processeurs de la machine. Afin de tester les deux modes d'exécution (séquentiel d'une part et distribué de l'autre), j'ai monté une expérience dont les résultats sont présentés dans le tableau 6.1. Pour celle-ci, les caractéristiques de 9 textes contenant de 10 à 3200 lignes ont été extraites et chronométrées.

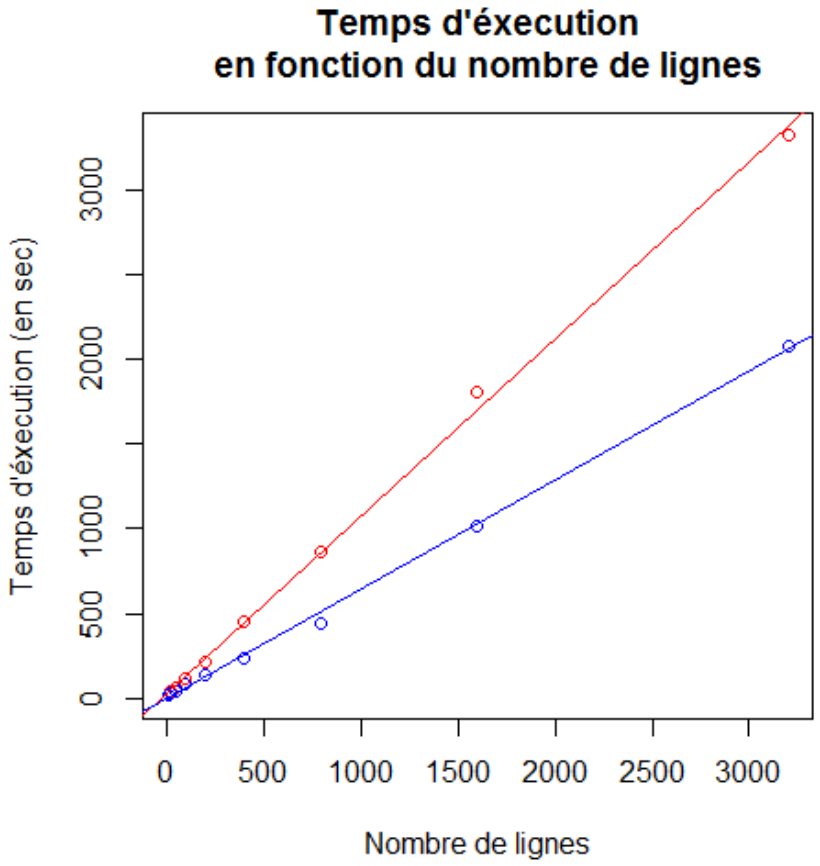


FIGURE 6.2: Comparaison des performances en temps d'exécution des modes distribué et séquentiel

Premièrement, on observe que la durée d'extraction des caractéristiques d'un texte est plus rapide en distribuant les lignes sur les processeurs qu'en mode séquentiel. De plus, plus le nombre de lignes est important plus l'écart de temps d'exécution entre les deux modes s'agrandit. Le graphique 6.2 présente les temps d'exécution des deux modes en fonction du nombre de lignes du texte. Les données du mode séquentiel sont en rouge alors que les données du mode distribué en bleu.

Grâce à ce graphique, on observe que les deux modes semblent suivre une distribution linéaire, ce qui est confirmé par les droites obtenues par régression linéaire (Seber and Lee, 2012). La droite dérivant le mode séquentiel a pour équation $Temps = 1.047 * nbLignes + 21,267$ et celle décrivant le mode distribué $Temps = 0.6433 * nbLignes + 2.9445$. Le temps d'exécution du mode parallèle est donc environ $1,63$ fois plus rapide que celui du mode séquentiel.

Ces résultats s'expliquent par le nombre de processeur de la machine sur laquelle

l'expérience a été réalisée. Le processeur de la machine est un Intel® Core™ i5-6300U[1] et contient 2 cœurs. La distribution des calculs s'est ainsi répartie sur ces deux cœurs, d'où la différence de facteur des régressions linéaires. J'en déduis que le verrou de l'analyse séquentielle est le temps moyen de traitement d'une phrase alors que le verrou de l'analyse parallèle est le nombre de cœurs de la (ou des) machine(s).

Enfin, je rappelle que distribuer l'extraction des caractéristiques pour quelques milliers de textes n'a pas de sens, cependant anticiper la réutilisation du programme pour d'autres données en a beaucoup, surtout dans le contexte actuel où le Big Data est générateur de plus en plus de projets. C'est pour cette raison que j'ai décidé de produire une application capable de passer à l'échelle en conservant le mode distribué tout le long du développement de mon programme.

6.2 Conception Technique

Pour tester et évaluer mes recherches, j'ai créé un programme à partir de zéro sans reprise d'existant. De ce fait, je n'avais aucune contrainte de développement spécifique mis à part la capacité de distribution des calculs pour les raisons explicitées dans la section précédente. Parmi les frameworks de calculs distribués présentés dans la section précédente, j'ai fait le choix d'utiliser Apache Spark pour ses promesses de rapidité (100 fois plus rapide que Hadoop d'après le site officiel[2]), ses librairies dédiées à l'apprentissage automatique et sa communauté grandissante de statisticiens.

Le programme est essentiellement composé de code personnel mais fait également appel à plusieurs librairies. Cette section présente dans un premier temps le langage de programmation choisi et l'environnement de développement sélectionné pour supporter le programme. Une seconde partie expose l'ensemble des librairies nécessaires au bon fonctionnement de celui-ci. Enfin, je présente et commente les méthodes du programme dans une troisième partie.

6.2.1 Langage et Environnement de Développement

Le choix du langage de programmation est le point de départ de toutes les nouvelles applications. Il en existe une multitude, chacun étant décrit par un ou plusieurs paradigmes (Impératif, déclaratif, fonctionnel, logique, orienté objet, etc.). Chaque communauté a

[1] Plus d'informations sur la page officielle : http://ark.intel.com/fr/products/88190/Intel-Core-i5-6300U-Processor-3M-Cache-up-to-3_00-GHz

[2] http://spark.apache.org/

ses langages favoris, celle de l'apprentissage automatique s'oriente facilement vers le langage R qui fait également office de logiciel (Team, 2013), le langage Python pour ses nombreuses librairies (Van Rossum, 2007), le C++ pour sa rapidité d'exécution (Stroustrup, 1995), le JAVA pour sa communauté active (Arnold et al., 2000) et plus récemment le Scala (Odersky et al., 2008).

J'ai fait le choix d'utiliser le langage Scala (dans sa version 2.11.8) pour plusieurs raisons que j'expose ci-après :

- Le framework Apache Spark propose trois APIS : JAVA, Python et Scala, restreignant le choix du langage. JAVA est un langage orienté objet très verbeux qui n'est pas optimisé pour le calcul parallèle. Python est un langage impératif, orienté objet et fonctionnel très utilisé par la communauté. L'un de ses atouts est également son principal inconvénient : il ne s'exécute pas sur la Java Virtual Machine (JVM), le rendant incompatible avec les nombreuses librairies Java. Enfin Scala est le programme le plus récent de la liste et est orienté objet et fonctionnel. Son nom signifie *Scalable Language*, ce qui signifie qu'il a été créé pour le passage à l'échelle des applications. De plus, Apache Spark est majoritairement codé en Scala, c'est principalement pour cette raison que j'ai fait le choix de ce dernier.
- Scala est un langage fonctionnel, orienté objet et tournant sur la JVM le rendant compatible avec l'ensemble des librairies JAVA. De plus, il est possible d'écrire un programme Scala en JAVA, rendant ce langage flexible et facile à prendre en main.
- Scala est proposé avec le moteur de production Simple Build-Tool (SBT) (Lacava et al.) permettant entre autres de gérer simplement les dépendances d'un projet et donc de faciliter son utilisation et son déploiement.

Scala est supporté par plusieurs logiciels de développement (IDE) dont les très connus Netbeans[3] et Eclipse[4], cependant c'est Intellij IDEA[5] produit par Jetbrains[6] qui s'est imposé comme l'IDE de référence. C'est logiquement que j'ai fait le choix d'utiliser Intellij IDEA.

6.2.2 Librairies Externes

Le programme développé fait appel à plusieurs librairies essentielles à son bon fonctionnement. Ci-après, je liste et décris l'ensemble de celles-ci :

[3] https://fr.netbeans.org/
[4] https://eclipse.org/ide/
[5] https://www.jetbrains.com/idea/
[6] https://www.jetbrains.com/

- Spark Core 1.5.0 (Spark, 2014). Cette librairie écrite en Scala permet la création, la modification, la manipulation et la suppression des RDDs, l'abstraction de données spécifique à Spark. Dans mon programme, Spark Core est utilisé pour la distribution des traitements d'extraction des caractéristiques des textes.

- Owlapi 4.0.2 (Horridge and Bechhofer, 2011). Cette API écrite en JAVA permet la création, la modification, la manipulation et la suppression d'ontologies au format owl. Celle-ci m'est utile pour la montée en mémoire de l'ontologie DO dans GraphX[7].

- Spark GraphX 1.5.0 (Xin et al., 2013). GraphX est une librairie écrite en Scala de Spark permettant de manipuler des RDDs au travers d'une abstraction graphe. Dans mon programme, je la combine avec owlapi pour la manipulation de l'ontologie DO.

- Stanford NLP 3.5.2 (Manning et al., 2014). Cette librairie écrite en JAVA propose différentes méthodes de traitement automatique de la langue. J'utilise ses fonctions pour l'extraction des caractéristiques des textes.

6.2.3 Présentation des Méthodes Principales

Dans cette sous partie, je présente les principales méthodes de mon programme (entrées et sorties) et propose leur pseudo-algorithme.

6.2.3.1 Extraction des Groupes Nominaux

L'extraction des caractéristiques est réalisée avec la librairie Spark Core et Stanford NLP, prend en paramètre le fichier contenant le texte à analyser, le *SparkContext* (objet nécessaire à l'utilisation de Spark) et retourne la liste des groupes nominaux décrivant ce dernier. Cette partie a été développée en collaboration avec le doctorant Jean Petit. Le pseudo-algorithme de cette méthode est le suivant :

- DEBUT
  - Lecture du fichier en paramètre à l'aide de la méthode *sc.textFile()* de Spark.
  - Distribution des traitements pour chacune des phrases à l'aide de la librairie Spark Core et de la méthode *map()*.
    - Découpages des phrases en unités lexicales.

[7] Plus d'informations sur cette combinaison est disponible à l'adresse `https://jcrisch.wordpress.com/2015/09/05/charger-une-ontologie-owl-dans-spark-graphx/`.

        * Étiquetage morpho-syntaxique (POS tagging) des unités lexicales.
        * Lémmatisation des unités lexicales.
        * Extraction des groupes nominaux.

- FIN

6.2.3.2 Création du Modèle

La création du modèle fait appel à la méthode d'extraction des caractéristiques, en conséquence, cette méthode utilise l'ensemble des librairies de cette dernière. La méthode prend en paramètre le chemin absolu vers le dossier contenant l'ensemble des fichiers, l'ontologie montée en mémoire dans GraphX et exporte en sortie le modèle au format XML (permettant notamment sa visualisation à partir d'un programme externe). Le pseudo-algorithme est le suivant :

- DEBUT
    - Ouverture du dossier à partir du chemin absolu en paramètre. Pour tous les fichiers contenus, faire :
        * Extraction des groupes nominaux.
        * Filtrage des groupes nominaux à l'aide de l'ontologie montée dans GraphX.
    - Calcul des estimations des probabilités des concepts extraits à l'aide des formules proposées par le classifieur naïf Bayésien.
    - Enrichissement du modèle (si souhaité)
    - Écriture du modèle au format XML.
- FIN

6.2.3.3 Classification d'un Nouveau Texte

Tout comme la création du modèle, la méthode de classification fait appel à la fonction d'extraction des caractéristiques et utilise donc l'ensemble de ses librairies. Cette méthode prend en paramètre le modèle calculé lors de la création du modèle, le fichier à labelliser et retourne la catégorie proposée. Le pseudo-algorithme est le suivant :

- DEBUT
    - Extraction des groupes nominaux.
    - Filtrage des groupes nominaux à l'aide des concepts du modèle.

  - Calcul des probabilités d'appartenance à chaque classe à l'aide du modèle.
  - Sélection de la catégorie ayant obtenu le meilleur score.

- FIN

6.2.3.4 Enrichissement du Modèle

L'enrichissement du modèle est la principale contribution de mes recherches. Cette étape prend en paramètre l'ontologie montée dans GraphX, le modèle d'apprentissage, d_{max}, r et retourne l'abstraction du modèle enrichi. Le pseudo-algorithme est le suivant :

- DEBUT
  - Extraction de l'ensemble des concepts potentiels.
  - Sélection des concepts à inclure dans le modèle enrichi. Cette sélection est réalisée par le parcours de l'ontologie de manière récursive.
  - Calcul des estimations des probabilités des nouveaux concepts
- FIN

Chapitre 7

Conclusion et Perspective

Dans ce chapitre, je conclus en présentant les différentes contributions (présentées dans (Risch et al., 2016)) apportées dans une première section puis je propose quelques perspectives dans une troisième.

7.1 Conclusion

Au cours de ce travail de thèse, j'ai proposé deux contributions pour la classification de textes représentés par des concepts d'une ontologie. La première est une méthode d'enrichissement des modèles de classification à l'aide des concepts et de la structure de l'ontologie de domaine associée et la seconde est une analyse du rôle des relations sémantiques pour l'enrichissement des modèles. Ces contributions sont fondées sur le classifieur naïf Bayésien, l'un des algorithmes probabilistes d'apprentissage automatique le plus utilisé dans le domaine de la classification de textes. Également, j'ai proposé une méthode de visualisation interactive des modèles de classification au travers du graphe de l'ontologie. Enfin, dans un souci de généricité des travaux au sein de l'entreprise m'ayant accueilli, j'ai intégré à mes travaux la contrainte du passage à l'échelle (scalabilité) via la distribution des calculs.

Ma première contribution a pour objectif l'amélioration des performances des modèles de classification de textes représentés par des concepts d'une ontologie par l'ajout d'une étape d'enrichissement. Celle-ci est réalisée une fois la création du modèle terminée permettant la réutilisation de l'ensemble des règles de ce dernier. L'enrichissement du modèle consiste à intégrer un ensemble de concepts de l'ontologie associée, non extraits lors de l'apprentissage afin de représenter le plus largement possible le domaine de connaissances de l'ontologie au sein du modèle. L'ajout d'un concept est intégré ou non

suivant le nombre de relations sémantiques le séparant du concept du modèle le plus proche. Ce nombre est l'unique paramètre nécessaire à la méthode. J'ai proposé deux mesures permettant le calcul des estimations des probabilités des *nouveaux concepts*. La première est une combinaison simple des estimations des probabilités de l'ensemble des concepts du modèle considérés comme proches de ceux-ci et la seconde est une extension de la première, intégrant la distance entre les concepts. Une expérience, fondée sur le corpus de textes labellisés Ohsumed et l'ontologie de domaine Disease Ontology (DO), a permis de montrer que l'enrichissement des modèles de classification permet d'améliorer les scores de ces derniers. De plus, elle a mis en évidence l'importance du paramètre de distance, les modèles enrichis obtenant les résultats les meilleurs avec une distance maximale autorisée de deux sauts de relations sémantiques.

Ma seconde contribution est une suite logique de la première. Celle-ci vise à analyser le rôle des relations sémantiques dans l'enrichissement des modèles et ainsi découvrir si certains types de relations dégradent les modèles ou au contraire les améliorent. Se basant sur les données utilisées pour la construction des modèles précédents, l'analyse a porté sur les relations hiérarchiques d'hyperonymie et d'hyponymie (uniques relations sémantiques de l'ontologie DO). J'ai monté une seconde expérience pour évaluer l'effet de ces types de relations sur l'enrichissement. Celle-ci a mis en évidence que la qualité des modèles enrichis par les hyperonymes dépendait essentiellement de la valeur du paramètre de distance utilisé. Plus celui-ci est grand, plus le modèle enrichi est dégradé. J'ai aussi observé que les modèles enrichis par les hyponymes n'ont pas d'effet remarquable sur la qualité des modèles. Cependant, en comparant ces résultats avec ceux obtenus dans l'expérience précédente, il apparaît que l'utilisation des deux types de relations pour l'enrichissement permet d'obtenir des modèles de meilleure qualité.

7.2 Perspectives

Les perspectives de ces travaux sont nombreuses et diverses. Tout d'abord, l'un des premiers travaux à venir est l'adaptation des méthodes proposées à d'autres algorithmes d'apprentissage automatique. Le classifieur Bayésien est reconnu par la communauté, cependant d'autres algorithmes tels que le réseau de neurones ou la machine à vecteurs de support sont aussi régulièrement cités par la communauté et permettraient peut-être d'obtenir de meilleurs résultats.

Une seconde perspective concerne l'aspect applicatif de la méthode et serait souhaitée par l'entreprise. Elle concerne le passage en temps réel de la classification tout en conservant la scalabilité de cette dernière.

Enfin, une troisième perspective est la poursuite du développement de la visualisation proposée pour analyser le modèle de classification au travers du graphe de l'ontologie. Visualiser un modèle de classification est un atout important qui permet notamment à la méthode des arbres de décision d'être encore régulièrement utilisée par la communauté malgré ses performances en dessous des autres méthodes.

Bibliographie

Kjersti Aas and Line Eikvil. Text categorisation: A survey, 1999.

Rodrigo Agerri, Xabier Artola, Zuhaitz Beloki, German Rigau, and Aitor Soroa. Big data for natural language processing: A streaming approach. *Knowledge-Based Systems*, 79:36–42, 2015.

Charu C Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer, 2012.

S Al-Harbi, A Almuhareb, A Al-Thubaity, MS Khorsheed, and A Al-Rajeh. Automatic arabic text classification. 2008.

Mihael Ankerst, Christian Elsen, Martin Ester, and Hans-Peter Kriegel. Visual classification: an interactive approach to decision tree construction. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–396. ACM, 1999.

Chidanand Apté, Fred Damerau, and Sholom M Weiss. Towards language independent automated learning of text categorization models. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 23–30. Springer-Verlag New York, Inc., 1994.

Ken Arnold, James Gosling, David Holmes, and David Holmes. *The Java programming language*, volume 2. Addison-wesley Reading, 2000.

Morton M. Astrahan, Mike W. Blasgen, Donald D. Chamberlin, Kapali P. Eswaran, Jim N Gray, Patricia P. Griffiths, W Frank King, Raymond A. Lorie, Paul R. McJones, and James W. Mehl. System r: relational approach to database management. *ACM Transactions on Database Systems (TODS)*, 1(2):97–137, 1976.

L Douglas Baker and Andrew Kachites McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103. ACM, 1998.

Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: an open source software for exploring and manipulating networks. *ICWSM*, 8:361–362, 2009.

Vladimir Batagelj and Andrej Mrvar. Pajek-program for large network analysis. *Connections*, 21(2):47–57, 1998.

Mohammed Benkhalifa, Abdelhak Mouradi, and Houssaine Bouyakhf. Integrating external knowledge to supplement training data in semi-supervised learning for text categorization. *Information Retrieval*, 4(2):91–113, 2001.

Paul N Bennett. Using asymmetric distributions to improve text classifier probability estimates. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 111–118. ACM, 2003.

Kristi L Berg, Tom Seymour, and Richa Goel. History of databases. *International Journal of Management & Information Systems (Online)*, 17(1):29, 2013.

Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Database theory—ICDT'99*, pages 217–235. Springer, 1999.

Bear Bibeault and Yehuda Kats. *jQuery in Action*. Dreamtech Press, 2008.

Stephan Bloehdorn and Andreas Hotho. Text classification by boosting weak learners based on terms and concepts. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 331–334. IEEE, 2004.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Dan Brickley and Ramanathan V Guha. Resource description framework (rdf) schema specification 1.0: W3c candidate recommendation 27 march 2000. 2000.

Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational linguistics*, 21(4):543–565, 1995.

Paula Carvalho, Luís Sarmento, Mário J Silva, and Eugénio De Oliveira. Clues for detecting irony in user-generated contents: oh...!! it's so easy;-). In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56. ACM, 2009.

William B Cavnar and John M Trenkle. N-gram-based text categorization. *Ann Arbor MI*, 48113(2):161–175, 1994.

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

Pimwadee Chaovalit and Lina Zhou. Movie review mining: A comparison between supervised and unsupervised classification approaches. In *Proceedings of the 38th annual Hawaii international conference on system sciences*, pages 112c–112c. IEEE, 2005.

Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36, 1976.

Edgar F Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

William W Cohen and Yoram Singer. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems (TOIS)*, 17(2):141–173, 1999.

George Copeland and David Maier. Making smalltalk a database system. In *ACM Sigmod Record*, volume 14, pages 316–325. ACM, 1984.

Antoine Cornuéjols and Laurent Miclet. *Apprentissage artificiel: concepts et algorithmes*. Editions Eyrolles, 2011.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20 (3):273–297, 1995.

Thomas M Cover and Peter E Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9, 2006.

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Transferring naive bayes classifiers for text classification. In *Proceedings of the national conference on artificial intelligence*, volume 22, page 540. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

Franca Debole and Fabrizio Sebastiani. Supervised term weighting for automated text categorization. In *Text mining and its applications*, pages 81–97. Springer, 2004.

Franca Debole and Fabrizio Sebastiani. An analysis of the relative hardness of reuters-21578 subsets. *Journal of the American Society for Information Science and technology*, 56(6):584–596, 2005.

Irene Díaz, José Ranilla, Elena Montañes, Javier Fernández, and Elías F Combarro. Improving performance of text categorization by combining filtering and support vector machines. *Journal of the American society for information science and technology*, 55 (7):579–592, 2004.

Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000.

Vladimir Dobrynin, David Patterson, Mykola Galushka, and Niall Rooney. Sophia: an interactive cluster-based retrieval system for the ohsumed collection. *IEEE Transactions on Information Technology in Biomedicine*, 9(2):256–265, 2005.

Kai-Bo Duan and S Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. In *International Workshop on Multiple Classifier Systems*, pages 278–285. Springer, 2005.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

AJ Duineveld, Roeland Stoter, MR Weiden, Bisente Kenepa, and VR Benjamins. Wondertools? a comparative study of ontological engineering tools. *International Journal of Human-Computer Studies*, 52(6):1111–1133, 2000.

Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155. ACM, 1998.

Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

David Flanagan. *JavaScript: the definitive guide*. " O'Reilly Media, Inc.", 2006.

Christopher Fox. A stop list for general text. In *ACM SIGIR Forum*, volume 24, pages 19–21. ACM, 1989.

April Fritz, Constance Percy, Andrew Jack, Kanagaratnam Shanmugaratnam, Leslie Sobin, D Max Parkin, and Sharon Whelan. *International classification of diseases for oncology*. Number Ed. 3. World Health Organization, 2000.

Keinosuke Fukunaga and Warren LG Koontz. Application of the karhunen-loeve expansion to feature selection and ordering. *IEEE Transactions on Computers*, 19(4): 311–318, 1970.

Johannes Fürnkranz. A study using n-gram features for text categorization. *Austrian Research Institute for Artifical Intelligence*, 3(1998):1–10, 1998.

Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. Learning by transduction. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 148–155. Morgan Kaufmann Publishers Inc., 1998.

John H Gennari, Mark A Musen, Ray W Fergerson, William E Grosso, Monica Crubézy, Henrik Eriksson, Natalya F Noy, and Samson W Tu. The evolution of protégé: an environment for knowledge-based systems development. *International Journal of Human-computer studies*, 58(1):89–123, 2003.

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *ACM SIGOPS operating systems review*, volume 37, pages 29–43. ACM, 2003.

Jennifer Golbeck, Gilberto Fragoso, Frank Hartel, Jim Hendler, Jim Oberthaler, and Bijan Parsia. The national cancer institute's thesaurus and ontology. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1):75–80, 2003.

Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Unsupervised and semi-supervised clustering: a brief survey. *A review of machine learning techniques for processing multimedia content, Report of the MUSCLE European Network of Excellence (FP6)*, 2004.

Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5):907–928, 1995.

Nicola Guarino. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology*, pages 139–170. Springer, 1997.

Nicola Guarino. Formal ontology and information systems. In *Proceedings of FOIS*, volume 98, pages 81–97, 1998.

Isabelle Guyon and André Elisseeff. An introduction to feature extraction. In *Feature extraction*, pages 1–25. Springer, 2006.

Ada Hamosh, Alan F Scott, Joanna S Amberger, Carol A Bocchini, and Victor A McKusick. Online mendelian inheritance in man (omim), a knowledgebase of human genes and genetic disorders. *Nucleic acids research*, 33(suppl 1):D514–D517, 2005.

Jing Han, E Haihong, Guan Le, and Jian Du. Survey on nosql database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, pages 363–366. IEEE, 2011.

Bhat S Harish, Devanur S Guru, and Shantharamu Manjunath. Representation and classification of text documents: A brief review. *IJCA, Special Issue on RTIPPR (2)*, pages 110–119, 2010.

Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *Unsupervised learning*. Springer, 2009.

Ivan Herman, Guy Melançon, and M Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on visualization and computer graphics*, 6(1):24–43, 2000.

William Hersh, Chris Buckley, TJ Leone, and David Hickam. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *SIGIR'94*, pages 192–201. Springer, 1994.

Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

Liangjie Hong and Brian D Davison. Empirical study of topic modeling in twitter. In *Proceedings of the first workshop on social media analytics*, pages 80–88. ACM, 2010.

Matthew Horridge and Sean Bechhofer. The owl api: A java api for owl ontologies. *Semantic Web*, 2(1):11–21, 2011.

Andreas Hotho, Steffen Staab, and Gerd Stumme. Ontologies improve text document clustering. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 541–544. IEEE, 2003.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

Anna Huang, David Milne, Eibe Frank, and Ian H Witten. Clustering documents using a wikipedia-based concept representation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 628–636. Springer, 2009.

Thorsten Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.

Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, pages 237–285, 1996.

Kamvar Kamvar, Sepandar Sepandar, Klein Klein, Dan Dan, Manning Manning, and Christopher Christopher. Spectral learning. In *International Joint Conference of Artificial Intelligence*. Stanford InfoLab, 2003.

Bhaskar Kapoor and Savita Sharma. A comparative study ontology building tools for semantic web applications. *International journal of Web & Semantic Technology (IJWesT)*, 1(3), 2010.

Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou. Ontology visualization methods—a survey. *ACM Computing Surveys (CSUR)*, 39(4):10, 2007.

Athanasios Kehagias, Vassilios Petridis, Vassilis G Kaburlasos, and Pavlina Fragkou. A comparison of word-and sense-based text categorization using several classification algorithms. *Journal of Intelligent Information Systems*, 21(3):227–247, 2003.

Alistair Kennedy and Diana Inkpen. Sentiment classification of movie reviews using contextual valence shifters. *Computational intelligence*, 22(2):110–125, 2006.

Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. Some effective techniques for naive bayes text classification. *IEEE transactions on knowledge and data engineering*, 18(11):1457–1466, 2006.

Graham Klyne and Jeremy J Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics, 2003.

Vandana Korde and C Namrata Mahender. Text classification and classifiers: A survey. *International Journal of Artificial Intelligence & Applications*, 3(2):85, 2012.

Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques, 2007.

Praful Koturwar, Sheetal Girase, and Debajyoti Mukhopadhyay. A survey of classification techniques in the area of big data. *arXiv preprint arXiv:1503.07477*, 2015.

Andrew Kusiak. Feature transformation methods in data mining. *IEEE Transactions on Electronics packaging manufacturing*, 24(3):214–221, 2001.

JungAe Kwak and Hwan-Seung Yong. Ontology matching based on hypernym, hyponym, holonym, and meronym sets in word net. *International Journal of Web & Semantic Technology*, 1(2):1–14, 2010.

Alessandro Lacava, Janek Bogucki, Aliaksandr Bedrytski, Matthew de Detrich, and Benjamin Neil. Simple build tool. *Professional Scala*, pages 45–62.

Wai Lam and Kwok-Yin Lai. A meta-learning approach for text categorization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 303–309. ACM, 2001.

Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th international conference on machine learning*, pages 331–339, 1995.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Jerome Y Lettvin, Humberto R Maturana, Warren S McCulloch, and Walter H Pitts. What the frog's eye tells the frog's brain. *Proceedings of the IRE*, 47(11):1940–1951, 1959.

David D Lewis. Feature selection and feature extraction for text categorization. In *Proceedings of the workshop on Speech and Natural Language*, pages 212–217. Association for Computational Linguistics, 1992a.

David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning*, pages 4–15. Springer, 1998.

David D Lewis and W Bruce Croft. Term clustering of syntactic phrases. In *Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 385–404. ACM, 1989.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5 (Apr):361–397, 2004.

David Dolan Lewis. *Representation and learning in information retrieval*. PhD thesis, University of Massachusetts, 1992b.

Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 375–384. ACM, 2009.

Carolyn E Lipscomb. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88(3):265, 2000.

Julie B Lovins. *Development of a stemming algorithm*. MIT Information Processing Group, Electronic Systems Laboratory Cambridge, 1968.

Shing-Hwa Lu, Ding-An Chiang, Huan-Chao Keh, and Hui-Hua Huang. Chinese text classification by the naïve bayes classifier and the associative classifier with multiple confidence threshold values. *Knowledge-based systems*, 23(6):598–604, 2010.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

Thomas Marill and D Green. On the effectiveness of receptors in recognition systems. *IEEE transactions on Information Theory*, 9(1):11–17, 1963.

Brian McBride. Jena: Implementing the rdf model and syntax specification. In *Proceedings of the Second International Conference on Semantic Web-Volume 40*, pages 23–28. CEUR-WS. org, 2001.

Andrew McCallum. Multi-label text classification with a mixture model trained by em. In *AAAI'99 workshop on text learning*, pages 1–7, 1999.

Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

David Sawyer McFarland. *CSS3: the missing manual*. " O'Reilly Media, Inc.", 2012.

Deborah L McGuinness and Frank Van Harmelen. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

Xiangrui Meng, Joseph Bradley, B Yuvaz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D Tsai, Manish Amde, and Sean Owen. Mllib: Machine learning in apache spark. *JMLR*, 17(34):1–7, 2016.

James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209:415–446, 1909.

Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Eric Mjolsness and Dennis DeCoste. Machine learning for science: state of the art and future prospects. *Science*, 293(5537):2051–2055, 2001.

Dunja Mladenic and Marko Grobelnik. Word sequences as features in text-learning. In *In Proceedings of the 17th Electrotechnical and Computer Science Conference (ERK98*. Citeseer, 1998.

Bruce Momjian. *PostgreSQL: introduction and concepts*, volume 192. Addison-Wesley New York, 2001.

Alessandro Moschitti and Roberto Basili. Complex linguistic features for text classification: A comprehensive study. In *European Conference on Information Retrieval*, pages 181–196. Springer, 2004.

Pio Nardiello, Fabrizio Sebastiani, and Alessandro Sperduti. Discretizing continuous attributes in adaboost for text categorization. In *European Conference on Information Retrieval*, pages 320–334. Springer, 2003.

Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *ACM SIGIR Forum*, volume 31, pages 67–73. ACM, 1997.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39 (2-3):103–134, 2000.

Chikashi Nobata, Nigel Collier, and Jun-ichi Tsujii. Automatic term identification and classification in biology texts. In *Proc. of the 5th NLPRS*, pages 369–374, 1999.

Martin Odersky, Lex Spoon, and Bill Venners. *Programming in scala*. Artima Inc, 2008.

Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. Mahout in action. 2012.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

Fuchun Peng and Dale Schuurmans. Combining naive bayes and n-gram language models for text classification. In *European Conference on Information Retrieval*, pages 335–350. Springer, 2003.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web*, pages 91–100. ACM, 2008.

Mark Pilgrim. *HTML5: up and running.* " O'Reilly Media, Inc.", 2010.

Eric Prud'Hommeaux and Andy Seaborne. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, volume 3, pages 616–623. Washington DC), 2003.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of machine learning research*, 5(Jan):101–141, 2004.

CJ van Rijsbergen. Information retrieval. *Department of Computing Science, University of Glasgow*, 1975.

Jean-Charles Risch, Jean Petit, and Francis Rousseaux. Ontology-based supervised text classification in a big data and real time environment. *CoDIT'16, New Territories of Information Workshop*, 2016.

S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. 1990.

Magnus Sahlgren and Rickard Cöster. Using bag-of-concepts to improve the performance of support vector machines in text categorization. In *Proceedings of the 20th*

international conference on Computational Linguistics, page 487. Association for Computational Linguistics, 2004.

Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

Minoru Sasaki and Hiroyuki Shinnou. Spam detection using text clustering. In *2005 International Conference on Cyberworlds (CW'05)*, pages 4–pp. IEEE, 2005.

Lynn Marie Schriml, Cesar Arze, Suvarna Nadendla, Yu-Wei Wayne Chang, Mark Mazaitis, Victor Felix, Gang Feng, and Warren Alden Kibbe. Disease ontology: a backbone for disease semantic integration. *Nucleic acids research*, 40(D1):D940–D946, 2012.

Daniel A Schult and P Swart. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, volume 2008, pages 11–16, 2008.

Sam Scott and Stan Matwin. Text classification using wordnet hypernyms. In *Use of WordNet in natural language processing systems: Proceedings of the conference*, pages 38–44, 1998.

George AF Seber and Alan J Lee. *Linear regression analysis*, volume 936. John Wiley & Sons, 2012.

Kent A Spackman, Keith E Campbell, and Roger A Côté. Snomed rt: a reference terminology for health care. In *Proceedings of the AMIA annual fall symposium*, page 640. American Medical Informatics Association, 1997.

Apache Spark. Apache spark^{TM}-lightning-fast cluster computing, 2014.

Donald F Specht. Probabilistic neural networks. *Neural networks*, 3(1):109–118, 1990.

Donald F Specht. A general regression neural network. *Neural Networks, IEEE Transactions on*, 2(6):568–576, 1991.

Michael Stonebraker. Sql databases v. nosql databases. *Communications of the ACM*, 53(4):10–11, 2010.

Michael Stonebraker and Dorothy Moore. *Object Relational DBMSs: The Next Great Wave*. Morgan Kaufmann Publishers Inc., 1995.

Michael Stonebraker and Lawrence A Rowe. *The design of Postgres*, volume 15. ACM, 1986.

Michael Stonebraker, Gerald Held, Eugene Wong, and Peter Kreps. The design and implementation of ingres. *ACM Transactions on Database Systems (TODS)*, 1(3): 189–222, 1976.

Bjarne Stroustrup. *The C++ programming language*. Pearson Education India, 1995.

Xiaomeng Su and Lars Ilebrekke. A comparative study of ontology languages and tools. In *International Conference on Advanced Information Systems Engineering*, pages 761–765. Springer, 2002.

R Core Team. R: A language and environment for statistical computing. 2013.

Roberto Toldo, Umberto Castellani, and Andrea Fusiello. A bag of words approach for 3d object categorization. In *International Conference on Computer Vision/Computer Graphics Collaboration Techniques and Applications*, pages 116–127. Springer, 2009.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

Adil Toumouh, Ahmed Lehireche, Dominic Widdows, and Mimoun Malki. Adapting wordnet to the medical domain using lexicosyntactic patterns in the ohsumed corpus. *AICCSA*, 6:1029–1036, 2006.

Anh-Phuc Trinh. Classification de texte et estimation probabiliste par machine à vecteurs de support. *Actes du troisième DÉfi Fouille de Textes*, page 77, 2007.

S. Tufféry. *Data Mining et Statistique Décisionelle. Quatrième édition*. TECHNIP, Paris, 2012.

Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

Larry Ullman. *PHP for the world wide web: visual quickstart guide*. Peachpit Press, 2004.

Guido Van Rossum. Python programming language. In *USENIX Annual Technical Conference*, volume 41, 2007.

Vladimir Vapnik. Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780, 1963.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

Lipo Wang. *Support vector machines: theory and applications*, volume 177. Springer Science & Business Media, 2005.

Pu Wang, Jian Hu, Hua-Jun Zeng, Lijun Chen, and Zheng Chen. Improving text classification by using encyclopedia knowledge. In *Seventh IEEE international conference on data mining (ICDM 2007)*, pages 332–341. IEEE, 2007.

Tom White. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

Erik Wiener, Jan O Pedersen, and Andreas S Weigend. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th annual symposium on document analysis and information retrieval*, volume 317, page 332. Citeseer, 1995.

W John Wilbur and Karl Sirotkin. The automatic identification of stop words. *Journal of information science*, 18(1):45–55, 1992.

Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

Reynold S Xin, Joseph E Gonzalez, Michael J Franklin, and Ion Stoica. Graphx: A resilient distributed graph system on spark. In *First International Workshop on Graph Data Management Experiences and Systems*, page 2. ACM, 2013.

Rui Xu and Donald Wunsch. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

Yiming Yang. Noise reduction in a statistical approach to text categorization. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM, 1995.

Yiming Yang. An evaluation of statistical approaches to text categorization. *Information retrieval*, 1(1-2):69–90, 1999.

Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.

Yiming Yang and John Wilbur. Using corpus statistics to remove redundant words in text categorization. *JASIS*, 47(5):357–369, 1996.

Lixia Yao, Anna Divoli, Ilya Mayzus, James A Evans, and Andrey Rzhetsky. Benchmarking ontologies: bigger or better? *PLoS Comput Biol*, 7(1):e1001055, 2011.

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

Masoumeh Zareapoor and KR Seeja. Feature extraction or feature selection for text classification: A case study on phishing email detection. *International Journal of Information Engineering and Electronic Business*, 7(2):60, 2015.

Hai-Tao Zheng, Bo-Yeong Kang, and Hong-Gee Kim. Exploiting noun phrases and semantic relationships for text document clustering. *Information Sciences*, 179(13): 2249–2262, 2009.

Nick Qi Zhu. *Data visualization with D3. js cookbook*. Packt Publishing Ltd, 2013.

Xiaojin Zhu. Semi-supervised learning literature survey. 2005.

Guowei Zu, Wataru Ohyama, Tetsushi Wakabayashi, and Fumitaka Kimura. Accuracy improvement of automatic text classification based on feature transformation. In *Proceedings of the 2003 ACM symposium on Document engineering*, pages 118–120. ACM, 2003.

Enrichissement des Modèles de Classification de Textes Représentés par des Concepts

La majorité des méthodes de classification de textes utilisent le paradigme du sac de mots pour représenter les textes. Pourtant cette technique pose différents problèmes sémantiques : certains mots sont polysémiques, d'autres peuvent être des synonymes et être malgré tout différenciés, d'autres encore sont liés sémantiquement sans que cela soit pris en compte et enfin, certains mots perdent leur sens s'ils sont extraits de leur groupe nominal. Pour pallier ces problèmes, certaines méthodes ne représentent plus les textes par des mots mais par des concepts extraits d'une ontologie de domaine, intégrant ainsi la notion de sens au modèle. Les modèles intégrant la représentation des textes par des concepts restent peu utilisés à cause des résultats peu satisfaisants. Afin d'améliorer les performances de ces modèles, plusieurs méthodes ont été proposées pour enrichir les caractéristiques des textes à l'aide de nouveaux concepts extraits de bases de connaissances. Mes travaux donnent suite à ces approches en proposant une étape d'enrichissement des modèles à l'aide d'une ontologie de domaine associée. J'ai proposé deux mesures permettant d'estimer l'appartenance aux catégories de ces nouveaux concepts. A l'aide de l'algorithme du classifieur naïf Bayésien, j'ai testé et comparé mes contributions sur le corpus de textes labéllisés Ohsumed et l'ontologie de domaine Disease Ontology. Les résultats satisfaisants m'ont amené à analyser plus précisément le rôle des relations sémantiques dans l'enrichissement des modèles. Ces nouveaux travaux ont été le sujet d'une seconde expérience où il est question d'évaluer les apports des relations hiérarchiques d'hyperonymie et d'hyponymie.

Mots-clés : Classification de Textes, Intelligence Artificielle, Mégadonnées, Apprentissage Automatique, Visualisation de Données

Improving text-classification models using the bag-of-concept paradigm

Most of text-classification methods use the "bag of words" paradigm to represent texts. However Bloahdom and Hortho have identified four limits to this representation: (1) some words are polysemics, (2) others can be synonyms and yet differentiated in the analysis, (3) some words are strongly semantically linked without being taken into account in the representation as such and (4) certain words lose their meaning if they are extracted from their nominal group. To overcome these problems, some methods no longer represent texts with words but with concepts extracted from a domain ontology (Bag of Concept), integrating the notion of meaning into the model. Models integrating the bag of concepts remain less used because of the unsatisfactory results, thus several methods have been proposed to enrich text features using new concepts extracted from knowledge bases. My work follows these approaches by proposing a model-enrichment step using a domain ontology, I proposed two measures to estimate to belong to the categories of these new concepts. Using the naive Bayes classifier algorithm, I tested and compared my contributions on the Ohsumed corpus using the domain ontology "Disease Ontology". The satisfactory results led me to analyse more precisely the role of semantic relations in the enrichment step. These new works have been the subject of a second experiment in which we evaluate the contributions of the hierarchical relations of hypernymy and hyponymy.

Keywords: Text Classification, Artificial Intelligence, Big Data, Machine Learning, Data Visualization

Discipline : INFORMATIQUE

Spécialité : INTELLIGENCE ARTIFICIELLE

Université de Reims Champagne-Ardenne
EA 3804 CReSTIC
UFR Sciences Exactes et Naturelles
Moulin de la Housse
51867 Reims