HAL
open science

Méthodes d'apprentissage interactif pour la classification des messages courts

Ameni Bouaziz

To cite this version:

Ameni Bouaziz. Méthodes d'apprentissage interactif pour la classification des messages courts. Autre [cs.OH]. COMUE Université Côte d'Azur (2015 - 2019), 2017. Français. NNT : 2017AZUR4039 . tel-01590468

HAL Id: tel-01590468

https://theses.hal.science/tel-01590468v1

Submitted on 19 Sep 2017

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

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

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H È S E

pour obtenir le titre de

Docteur en Sciences

de l'Université Côte d'Azur

Mention : Informatique

Présentée et soutenue par

Ameni Bouaziz

Méthodes d'apprentissage interactif pour la classification des messages courts

Thèse dirigée par Frédéric Precioso

préparée à l'I3S Sophia Antipolis

soutenue le 19/06/2017

Jury :

Rapporteurs :	Laurent HEUTTE	-	Professeur, Univ. de Rouen
	Mathieu ROCHE	-	Chercheur HDR, Univ. Montpellier
Directeur :	Frédéric Precioso	-	Professeur, Univ. Côte d'Azur
Président :	Michael KRAJECKI	-	Professeur, Univ. de Reims Champagne Ardenne
Examinateur :	Dario MALCHIODI	-	Professeur, Univ. de Milan
Encadrants :	Célia da Costa Pereira	-	Maître de conférences, Univ. Côte d'Azur
	Christel Dartigues-Pallez	-	Maître de conférences, Univ. Côte d'Azur
Invité :	Philippe Van Den Bulke	-	Président de Semantic Groupig Company

A ton âme,
ma chère maman.
Tu me manques ...

Remerciements

Au terme de ma thèse je tiens à remercier chaleureusement mon directeur de thèse Frédéric Precioso et mes deux co-encadrantes Célia da Costa Pereira et Christel Dartigues-Pallez sans qui ce travail n'aura pas pu être réalisé. Je leur exprime mon entière gratitude pour tout le temps qu'ils m'ont consacré et pour leurs conseils précieux qui m'ont permis d'avancer tout le long de ma thèse. Je les remercie aussi de m'avoir initiée au monde de la recherche scientifique et pour tout ce que j'ai appris d'eux en étant d'abord leur étudiante, puis leur stagiaire et enfin leur doctorante. Je n'oublierai jamais également leurs qualités humaines que j'ai pu découvrir durant cette thèse.

Je tiens à remercier ensuite le professeur Laurent Heutte de l'université de Rouen et Monsieur Mathieu Roche, chercheur HDR à l'université Montpellier d'avoir accepté de rapporter cette thèse. Je remercie également le professeur Michael Krajecki de l'université de Reims Champagne Ardenne et le professeur Dario Malchiodi de l'université de Milan d'avoir accepté de faire partie du jury de ma soutenance. J'adresse aussi mes remerciements à Philippe Van Den Bulke, président de Semantic Grouping Company et également membre du jury pour avoir cofinancé ma thèse.

Un grand merci à toute l'équipe administrative du laboratoire I3S et d'une façon particulière à Magali Richir pour avoir préparé mes missions et pour m'avoir accompagnée dans toutes mes démarches administratives

Je remercie également tous les membres de l'équipe SPARKS du laboratoire I3S et en particulier ceux de l'équipe MinD que j'avais le plaisir de coutoyer durant ces années de thèse pour leur amitié.

Mes remerciements s'adressent aussi à mon cher père pour avoir toujours cru en moi, pour son encouragement continu et son soutien dans toutes mes décisions. Merci infiniment pour son grand amour et sa veille sur moi et sur mes études. Je remercie également ma belle mère Lamia, mes charmantes sœurs Ines, Douha et Eya ainsi que ma belle famille pour tout leur soutien et amour.

Anis, mon cher mari, je te remercie de tout mon cœur d'être mon ange gardien. Tu es toujours là, à mes côtes, pour me soutenir et pour m'encourager à avancer dans ce que je fais. Merci pour ta patience et ton amour.

Je clôture cette section par un spécial remerciement à mon petit ange Yasmine que j'ai eu la chance d'avoir pendant cette thèse, d'être toujours ma source de bonheur et d'amour. Je te souhaite ma chère fille plein de bonheur et de réussite dans toute ta vie.

Cette thèse a été financée par la région Provence Alpes Côte d'Azur (PACA) et l'entreprise Semantic Grouping Company (SGC)

Table des matières

1 Introduction générale **1**

2 Processus de classification des messages courts : État de l'art **5**
2.1 Introduction 5
2.2 Caractéristiques des messages courts 7
2.3 Prétraitement des messages courts 8
2.3.1 Topics Models 8
2.3.2 Enrichissement des messages courts 15
2.3.3 Réduction des messages courts 25
2.4 Apprentissage des messages courts 33
2.4.1 Support Vector Machines 33
2.4.2 Entropie Maximum 35
2.4.3 Naive Bayes 37
2.5 Classification des messages courts 37
2.6 Validation de la classification des messages courts 40
2.6.1 Accuracy 40
2.6.2 Erreur de classification 40
2.6.3 F1-mesure 41
2.6.4 Validation croisée 41
2.7 Synthèse 42

3 Forêts aléatoires **45**
3.1 Introduction 45
3.2 Arbres de décision 46
3.3 Bagging 50
3.4 Random Feature Selection 51
3.5 Force et corrélation 52
3.6 Consistance des forêts aléatoires 53
3.7 Types de forêts aléatoires 55
3.8 Forêts aléatoires et apprentissage dynamique 56
3.9 Synthèse 61

4 Forêts Sémantiques **63**
4.1 Introduction 63
4.2 Enrichissement 65
4.2.1 Enrichissement niveau mot 68
4.2.2 Enrichissement niveau message 68
4.3 Introduction de la sémantique dans les forêts aléatoires 71
4.4 Expérimentations et résultats 74
4.4.1 Benchmark 74

4.4.2 Résultats et interprétations 75
4.5 Tests statistiques 81
4.6 Synthèse 85

5 IGLM : Méthode d'apprentissage interactif et générique pour la classification des messages courts 87
5.1 Introduction 87
5.2 Abstraction 89
5.3 Condition de mise à jour du modèle 91
5.4 Adaptation : mise à jour du modèle de classification 93
5.5 Expérimentations et résultats 94
5.5.1 Protocole expérimental 94
5.5.2 Résultats et analyses 95
5.6 Synthèse 100

6 Étude comparative des forêts sémantiques et IGLM 103
6.1 Introduction 103
6.2 Points communs et points de différence entre les forêts sémantiques et IGLM 104
6.2.1 Points communs 104
6.2.2 Points de différence 104
6.3 Accuracy 105
6.4 Taille des Forêts 108
6.5 Temps de calcul et complexité 110
6.5.1 Temps de calcul et complexité de l'enrichissement et de l'abstraction 110
6.5.2 Temps de calcul des algorithmes d'apprentissages 112
6.6 Parcimonie ou "Sparsity" 113
6.7 Discrimination des caractéristiques 116
6.8 Synthèse 118

7 Conclusion générale 123
7.1 Conclusion 123
7.2 Perspectives 125
7.3 Conclusion et perspectives pour "Semantic Grouping Company" . . . 127

Bibliographie 129

Table des figures

2.1 Processus de classification des messages courts. 6
2.2 Un exemple d'un corpus simple constitué de 9 titres de documents techniques et la matrice associée [26]. 9
2.3 Représentation géométrique des documents et termes dans un espace de 2 dimensions [26]. 10
2.4 Paramétrisation asymétrique et symétrique du modèle d'aspects [35]. 11
2.5 Représentation graphique du modèle de LDA [13]. 13
2.6 Plate-forme générale proposée par Hu et Zhang dans [40] pour améliorer le regroupement de textes en introduisant la sémantique externe par Wikipédia. 16
2.7 Illustration de la correspondance entre les textes, les concepts et les catégories de Wikipédia dans [40]. 17
2.8 Algorithme de génération de caractéristiques extérieures proposé par Hu et Sun dans [39]. 19
2.9 Plate-forme générale proposée par Phan pour l'apprentissage des messages courts avec les thèmes cachés [59]. 20
2.10 Plate-forme de multi-granularité proposée par Chen et al. dans [21]. . 21
2.11 Plate-forme générale proposée par Tang et al. pour le regroupement de messages courts en utilisant la connaissance multi-langues [77]. . . 23
2.12 Algorithme de l'abstraction proposé par Silvescu [72]. 26
2.13 Algorithme de relief proposé par Kira et Redell dans [43]. 32
2.14 les SVM : l'hyperplan optimal séparant linéairement les deux nuages de points [5]. 34
2.15 Plate-forme proposée par Sun dans [76] pour la classification des messages courts. 38

3.1 Matrice représentative des 9 titres de documents [26]. 46
3.2 Modèle d'arbre généré à partir de la base de titres. 47
3.3 Illustration du bagging sur la base de titres. 50
3.4 Exemple de bootstrap ainsi que l'arbre généré sur la base des titres. . 51
3.5 Illustration du RFS sur la base de titres. 52
3.6 Les scénarios de distribution des données d'apprentissage [1]. 57
3.7 Exemple d'un arbre d'options [58]. 58

4.1 Schéma général des forêts sémantiques. 64
4.2 Algorithme d'enrichissement. 67
4.3 Exemple de base avant enrichissement. 69
4.4 Exemple de base après enrichissement. 70
4.5 Réseau sémantique construit par LDA. 72
4.6 Forêts sémantiques au niveau nœud 75

4.7 Variation de l'accuracy de la classification des messages courts en variant le nombre de thèmes ajoutés par message. 76
4.8 Variation de l'accuracy en fonction du nombre d'arbres en utilisant les forêts traditionnelles, les arbres sémantiques et les arbres sémantiques niveau nœuds. 80

5.1 Mécanisme général de IGLM. 89
5.2 Illustration du processus d'abstraction dans IGLM. 91
5.3 Matrice $\mathbf{M}_{(m\times n')}$ et matrice $\mathbf{M}_{(m+k\times n')}$ après mise à jour de la matrice $\mathbf{M}_{(m\times n')}$ avec k textes mal classés. 92
5.4 Illustration du processus d'adaptation dans IGLM. 94
5.5 Protocole expérimental de IGLM. 96
5.6 Résumé des résultats obtenus pour les différentes étapes du protocole. 97
5.7 Evolution de l'accuracy dans IGLM pour les 4 benchmarks. 99
5.8 Évolution des valeurs d'epsilon et de Δ 102

6.1 Variation de l'accuracy en fonction du nombre d'arbres pour l'enrichissement, l'abstraction et les forêts traditionnelles. 106
6.2 Erreur de classification de l'enrichissement et de l'abstraction par catégorie. 107
6.3 les mesures de parcimonie [41]. 115
6.4 Illustration de l'étude de la qualité de la matrice représentative des messages courts. 117

Chapitre 1

Introduction générale

L'apparition du web 2.0 a permis aux utilisateurs d'Internet de passer du statut de simples usagers récepteurs d'information à des internautes actifs qui interagissent avec le Web pour s'exprimer, donner leurs points de vue et communiquer. Sur les réseaux sociaux tels que Twitter et Facebook, un utilisateur peut poster des publications ou en commenter d'autres. Sur les sites de commerce électronique, un acheteur a la possibilité de donner son opinion sur un produit ou un service. Internet constitue donc un espace ouvert où tout le monde peut s'exprimer produisant ainsi des quantités importantes de données. Ces dernières sont sous des formats divers et variés, cependant, une grande partie est constituée de messages courts. Le Web 2.0 n'est qu'un exemple de sources fournissant les messages courts. On les trouve également dans les SMS envoyés par les téléphones portables ou dans les requêtes soumises au moteurs de recherche. Ce type de données est très riche en information mais n'est pas toujours structuré. Il peut jouer un rôle très important dans l'aide à la prise de décisions stratégiques dans plusieurs domaines si leur connaissance intrinsèque est correctement extraite. Pour ce faire, les techniques de fouille de textes sont employées, parmi lesquelles la classification automatique des messages. Celle-ci peut être utilisée, par exemple, pour la détection des spams dans les SMS. Un autre domaine très connu dans lequel la classification des messages courts joue un rôle important depuis le début des années 2000 est l'analyse des sentiments ou "Sentiment Analysis". Ce domaine consiste à dégager un sentiment à partir d'un message, généralement à partir des tweets. Un message peut ainsi être classé comme positif pour exprimer la satisfaction, négatif pour exprimer le mécontentement ou neutre. Les sentiments extraits à partir des messages peuvent ensuite faire l'objet de statistiques sur le ressenti général d'une communauté : ils peuvent par exemple donner une idée sur le taux de satisfaction des clients d'un produit particulier ou servir de sondage pour anticiper le succès d'un nouveau produit tel qu'un film ou le résultat des élections politiques.

Dans un cadre plus spécifique, notre partenaire industriel "Semantic Grouping Company" travaille sur un outil appelé "Meeting Software" permettant d'analyser et de regrouper les messages courts. Cet outil utilise différentes techniques de traitement automatique de la langue (TAL) et de la fouille des textes dont la classification automatique. Le but de l'outil est d'améliorer la performances des réunions professionnelles telles que les séminaires en profitant de l'échange collaboratif entre les participants aux réunions. Ainsi, une personne qui assiste à un séminaire ne se contente pas d'écouter le présentateur mais elle intervient également pour donner son avis en répondant à des enquêtes en ligne menées par "Meeting Software". Avant

chaque séminaire, un ensemble de questions liées au thème discuté est préparé. Lors du séminaire, ces questions sont posées aux participants qui possèdent des tablettes pour soumettre leurs réponses. Ces dernières sont des textes courts qui ne dépassent pas un certain nombre de caractères. Pour chaque question, les techniques de regroupement automatique ("clustering") sont appliquées sur une première collecte de données. Le résultat de cette phase, qui est un ensemble de réponses groupées par catégorie, est utilisé pour apprendre un modèle de classification. Ce dernier est utilisé ensuite pour classer les nouvelles données arrivant au cours du temps. Un pilote intervient pour corriger les erreurs potentielles de classification du modèle créé. Une fois que les données sont classées, les résultats obtenus ainsi que des statistiques tirées à partir d'eux sont affichés aux participants. Ces statistiques sont utiles pour analyser les données et aider à prendre les bonnes décisions.

Étant donné que la classification automatique des messages courts est une spécialisation de la fouille de textes, la procédure suivie est la même que celle réalisée lors de la classification des textes traditionnels. Cependant, des particularités caractérisent ce type de textes et rendent leur classification plus difficile. En effet, leur courte taille, leur faible densité ainsi que leur manque de contexte rendent les algorithmes d'apprentissage classiques moins performants que lorsqu'ils sont appliqués dans le cadre des textes longs. Une mauvaise classification engendre une intervention manuelle fréquente d'un pilote pour corriger les erreurs de classification. Cette intervention n'est pas pratique et est coûteuse en terme de temps. Le but de cette thèse est de la réduire en améliorant le plus possible la performance des modèles de classification générés. En s'appuyant sur l'état de l'art de la fouille des messages courts, on constate la présence d'une phase préliminaire précédant la création du modèle de classification. Elle consiste à améliorer la qualité des textes généralement par le biais de la sémantique pour atténuer l'effet de leur faible densité et leur manque de contexte. Cette phase permet d'améliorer la performance des algorithmes d'apprentissage utilisés en les aidant à mieux détecter la similarité entre les messages.

Le choix de l'algorithme d'apprentissage adapté joue un rôle important dans la performance de la classification des messages courts. Plusieurs types d'algorithmes existent dans la littérature. La tendance se dirige vers une famille d'algorithmes d'apprentissage supervisé développés ces dernières années, il s'agit de l'ensemble de classifieurs (EoC). Leur principe général consiste à utiliser plusieurs classifieurs de même type au lieu d'un seul sur la même base d'apprentissage. La prédiction finale est alors déduite à partir de la prédiction de chaque classifieur. Plusieurs méthodes d'induction d'un ensemble de classifieurs existent dans la littérature, les plus populaires sont le bagging et le boosting. Nous nous intéressons dans cette thèse à un algorithme qui a été utilisé avec succès dans plusieurs domaines notamment dans la fouille de textes : les forêts aléatoires. Cet algorithme consiste à combiner un ensemble d'arbres de décision au sein d'un seul classifieur où chaque arbre a son propre vote et où la décision finale de la forêt est le résultat d'un vote majoritaire. Il a été démontré dans la littérature que le classifieur arbre de décision en lui même n'est pas le plus performant et ceci à cause de sa forte dépendance aux données

d'apprentissage ainsi de sa forte probabilité de tomber dans le sur-apprentissage. Les forêts aléatoires sont conçues pour remédier à ces limites en se basant sur le principe que l'avis de plusieurs experts est meilleur que l'avis d'un seul et en introduisant un haut niveau de diversité dans le processus de construction de chaque arbre, un facteur important pour réduire l'instabilité. Cette diversité est introduite sur deux niveaux : au niveau des données par le principe de bagging et au niveau des caractéristiques par le principe de "Random Feature Selection". Il a été démontré que les forêts aléatoires ne souffrent pas de sur-apprentissage puisque l'erreur converge quand le nombre d'arbres tend vers l'infini. Cet algorithme d'apprentissage possède également plusieurs caractéristiques intéressantes telles que la nature de sa construction basée sur une segmentation sémantique de données. C'est ce qui a orienté notre choix sur cet algorithme comme une base pour nos travaux sur la classification des messages courts.

Deux grandes pistes d'amélioration de la classification des messages courts sont identifiées :

— introduire de la sémantique dans le processus de classification. En effet, sans tenir compte de la sémantique des messages et vu leur courte taille qui n'engendre pas assez de mots communs entre les messages, les méthodes traditionnelles calculant la similarité entre deux messages qui ne possèdent aucun mot en commun n'arrivent pas à les classer dans la même catégorie, alors qu'en réalité, le contexte de ces deux messages est le même mais il est juste exprimé différemment. Une méthode permettant de créer des liens sémantiques entre les messages est nécessaire ;

— proposer une méthode interactive qui profite de l'expertise de l'utilisateur lors de son intervention pour contrôler la classification automatique et corriger ses erreurs. La qualité du modèle de classification sera améliorée grâce à cette intervention qui en contre partie diminue au cours du temps.

Pour répondre à ces objectifs, nous proposons dans cette thèse deux nouvelles approches dédiées à la classification des messages courts tout en se basant sur les forêts aléatoires pour apprendre les différents modèles de classification. Nous commençons dans le chapitre suivant par proposer une étude de l'état de l'art du processus complet de la classification des textes courts. Nous présentons les différentes solutions proposées pour contourner les limites de ce type de textes ainsi que les différentes méthodes utilisées dans les étapes d'apprentissage, de classification et de validation. Le chapitre 3 rappelle le principe des forêts aléatoires ainsi que son état de l'art. Le chapitre 4 propose notre première contribution, une nouvelle approche qui intègre la sémantique dans différents niveaux du processus de classification des messages en offrant une méthode permettant d'améliorer la qualité des messages courts dans un premier temps et un nouveau type de forêts aléatoires qui tient compte de la sémantique des messages dans son comportement. Nous proposons ensuite une étude expérimentale permettant de montrer l'apport de notre méthode par rapport aux algorithmes traditionnels d'apprentissage. Le chapitre 5 décrit notre deuxième contribution dans le domaine de la fouille des textes courts où un processus interactif est proposé permettant de mettre à jour de façon continue les forêts aléatoires pour

améliorer leur performance au cours du temps tout en se basant sur des propriétés bien définies pour implanter dynamiquement les arbres de la forêt. Un protocole expérimental a été élaboré pour valider cette nouvelle approche et tracer l'évolution de la classification au cours du temps. Nous proposons ensuite dans le chapitre 6 une étude comparative détaillée de ces deux contributions sur plusieurs axes. Nous finissons cette thèse par une conclusion générale résumant nos travaux et proposant des suites possibles à ces contributions.

Chapitre 2

Processus de classification des messages courts : État de l'art

Sommaire

2.1 Introduction **5**
2.2 Caractéristiques des messages courts **7**
2.3 Prétraitement des messages courts **8**
2.3.1 Topics Models 8
2.3.2 Enrichissement des messages courts 15
2.3.3 Réduction des messages courts 25
2.4 Apprentissage des messages courts **33**
2.4.1 Support Vector Machines 33
2.4.2 Entropie Maximum 35
2.4.3 Naive Bayes 37
2.5 Classification des messages courts **37**
2.6 Validation de la classification des messages courts **40**
2.6.1 Accuracy 40
2.6.2 Erreur de classification 40
2.6.3 F1-mesure 41
2.6.4 Validation croisée 41
2.7 Synthèse **42**

2.1 Introduction

Comme outil de communication à la fois rapide et efficace, les messages courts sont de plus en plus présents de nos jours. Ils peuvent l'être sous la forme de SMS envoyés par les téléphones portables, des commentaires sur les réseaux sociaux ainsi que sur les sites de commerce électronique...
Ces messages représentent une quantité énorme de données qui peut être très utile pour prendre les bonnes décisions dans différents domaines pour, par exemple, améliorer la qualité d'un produit ou encore analyser un climat politique. Classiquement, les techniques de fouille de données sont utilisées pour exploiter la connaissance intrinsèque des messages. Cette thèse s'intéresse à la classification automatique de ces messages.

Ce type de classification est un processus de transformation des messages courts à partir d'un ensemble de données brutes vers un modèle de classification capable de prédire les classes des nouveaux messages avec une certaine probabilité de réussite. Ce chapitre passe en revue l'état de l'art de ce processus qui peut être résumé par la figure 2.1.

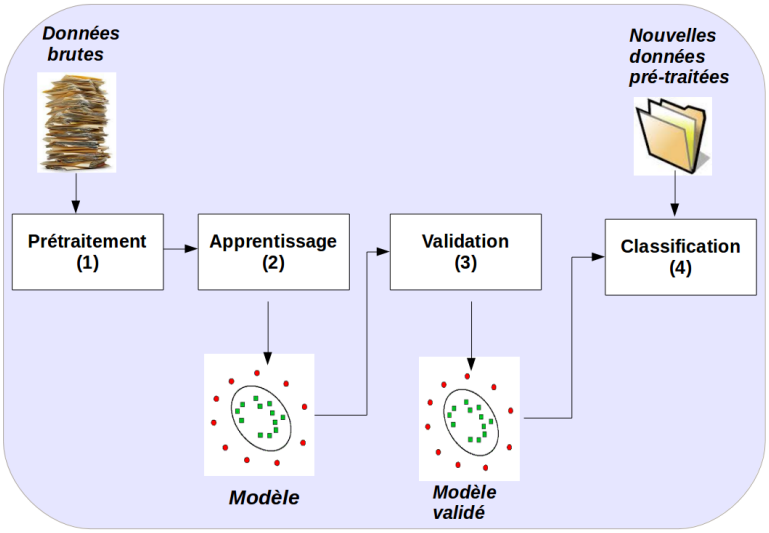


FIGURE 2.1 – Processus de classification des messages courts.

Le processus de classification des messages courts est composé de 4 phases :

1. **Phase de prétraitement** :
   Cette phase consiste à transformer les données brutes en données exploitables et compréhensibles par la machine. Il s'agit de les nettoyer, de les mettre en forme et de traiter le cas de données manquantes par exemple en créant de nouvelles données. Cette phase est très importante car la qualité du modèle créé dépend directement de ses résultats. Dans la fouille des textes, cette tâche est réalisée à l'aide de l'utilisation des outils de Traitement Automatique de la Langue (TAL) qui permettent de découper du texte en mots, de transformer des mots en leurs racines (lemmatisation), d'éliminer des mots non significatifs ou mots d'arrêt...
   Le résultat de cette phase est une représentation matricielle des données d'apprentissage qu'on appelle le sac des mots ou le "Bag Of Words "(BOW) [91]. Chaque ligne de cette matrice est un vecteur représentant un message. Chaque colonne représente un mot parmi tous les mots collectés dans le BOW. Ce mot est appelé une caractéristique. La valeur de chaque case est calculée à partir d'un poids se basant sur l'appartenance d'un mot à un message.
   Dans le cas des messages courts la représentation traditionnelle des données (BOW) est améliorée par plusieurs mécanismes comme l'enrichissement des messages ou la réduction de l'espace des caractéristiques.

2. **Phase d'apprentissage**
   Cette phase consiste à construire un modèle à partir de la matrice obtenue lors de la phase de prétraitement. Plusieurs familles d'apprentissage existent. Nous en distinguons deux principaux : L'apprentissage supervisé où le modèle est construit à partir d'un ensemble de données labellisées et l'apprentissage non supervisé qui consiste à diviser l'ensemble de données de départ en des groupes non connus par avance. Cette thèse s'intéresse à l'apprentissage supervisé puisque nous sommes dans un contexte de classification de messages courts dans des groupes prédéfinis. Il existe plusieurs algorithmes d'apprentissage supervisés dont certains ont été utilisés pour la classification des messages courts (SVM [25]) ou Maximum Entropy [7]). Aucun de ces algorithmes n'est à priori meilleur que les autres. C'est la nature des données d'apprentissage qui permet d'orienter le choix de la meilleure technique à utiliser.
3. **Phase de validation**
   Cette phase consiste à évaluer et valider le modèle créé et ceci par l'utilisation de mesures d'évaluation. L'implémentation de cette phase se fait en appliquant le modèle obtenu au cours de la phase précédente sur des données (messages) pour lesquelles l'information qui peut en être extraite est connu au préalable, puis de comparer les résultats obtenus à ceux attendus. Après cette phase le modèle sera prêt à être utilisé sur des données réelles. Plusieurs méthodes d'évaluation ont été appliquées sur les modèles de classification des messages courts comme l'accuracy et la validation croisée.
4. **Phase de classification**
   Cette phase consiste à exploiter en situation réelle le modèle créé et validé lors des phases précédentes. Une fois que le modèle est validé, on l'utilise pour classer les nouvelles données qui sont, dans notre cas, les nouveaux messages arrivant au cours du temps et pour lesquels nous ne connaissons pas cette fois la classe à prédire.

2.2 Caractéristiques des messages courts

La classification des messages courts pose un défi additionnel par rapport à la classification des textes traditionnels. Ceci est principalement dû aux caractéristiques de ces messages qui ont été étudiées dans différents travaux de la littérature. Elles sont résumées dans les points suivants :

— **courte taille** : en effet le message est constitué généralement d'une dizaine de mots. Un tweet par exemple ne dépasse pas 140 caractères. A cause de leur courte taille, les messages ne peuvent pas fournir un nombre suffisant d'occurrences de mots, rendant ainsi l'utilisation de mesures de similarité ainsi que l'identification du contexte plus difficile [59], [77], [39], [85], [74], [76], [82].

— **parcimonie** [67] : la matrice représentative des messages courts est une

matrice creuse, car les vecteurs représentant les messages sont des vecteurs très longs mais vides. En effet la taille d'un vecteur est égale au nombre total de tous les mots de l'ensemble des messages courts et comme un message n'est constitué que de quelques mots, on se trouve avec un vecteur qui contient une majorité de valeurs nulles. La parcimonie, ou la faible densité de la matrice représentative des messages, engendre un modèle de classification peu efficace. Il s'agit en effet du problème le plus important de ce type de messages. Il été évoqué par la plupart des chercheurs travaillant dans ce domaine tels que [59], [21], [39], [63], [73], [74], [82].

— **bruit** : que ce soient des commentaires sur les réseaux sociaux, des requêtes envoyées aux moteurs de recherche ou des SMS, les messages courts sont généralement des textes qui ont été écris rapidement par les utilisateurs. Il est donc fréquent qu'ils contiennent des fautes de frappe, des erreurs d'orthographe, des écritures télégraphiques, des émoticônes et des mots abrégés [77], [78]. Les textes courts sont généralement des données bruitées, ce qui dégrade la qualité des modèles de classification [59].

2.3 Prétraitement des messages courts

Cette phase consiste à transformer l'ensemble brut des messages en une matrice compréhensible par les algorithmes d'apprentissage. C'est dans cette phase que les chercheurs agissent en améliorant la qualité des messages. Nous avons identifié deux types de travaux : l'enrichissement des messages et la réduction de leur espace de caractéristiques (mots). Plusieurs techniques d'enrichissement et de réduction existent dans la littérature, la plupart se basent sur la notion des "Topics Models".

2.3.1 Topics Models

Les techniques du Topics Models sont utilisées dans le domaine de l'apprentissage automatique et dans le domaine de traitement automatique de la langue. Elles consistent généralement à découvrir des thèmes cachés dans un ensemble de documents. Plusieurs implémentations ont été réalisées, les plus connues étant LSI (ou LSA) [26], PLSI (ou PLSA) [36] et LDA [13].

2.3.1.1 Latent Semantic Analysis (LSA) ou Latent Semantic Indexing (LSI)

LSI ou LSA, introduite par Derrwester et al. dans [26], est une technique conçue principalement pour le domaine de recherche d'information. Elle consiste à capter la structure sémantique cachée dans les documents textuels en associant à un terme un ensemble de concepts qui lui sont reliés, ce qui mène à résoudre quelques problèmes de synonymie et polysémie qui ne sont pas détectables si nous prenons les termes tous seuls [57]. La clé de la réussite de cette méthode se trouve donc dans sa capacité à établir des liens entre les termes qui apparaissent dans un même contexte.

Le mécanisme de LSI se base sur les techniques d'algèbre linéaire. Il transforme la dimension de la représentation matricielle traditionnelle du corpus des données en une autre plus réduite en utilisant la décomposition en valeurs singulières (SVD). Soit $A_{(n,m)}$ la matrice qui contient n termes et m documents et de rang r. La SVD de A sera donc :

$$A = U_{(n,r)} \times D_{(r,r)} \times V^T_{(r,m)} \tag{2.1}$$

où : $D_{(r,r)}$ est une matrice diagonale et $U_{(n,r)}$ et $V_{(m,r)}$ sont deux matrices dont les colonnes sont orthogonales.

Une fois que la matrice initiale A est décomposée en trois sous matrices, LSI redimensionne la décomposition en ne gardant que les k termes les plus intéressants. Le choix de k représente un compromis. En effet, ce nombre doit être à la fois le plus petit possible afin de ne pas ralentir le mécanisme de recherche d'information et suffisamment large pour capter la structure entière de la sémantique du corpus. Nous obtenons ainsi :

$$A_k = U_k \times D_k \times V_k^T \tag{2.2}$$

La représentation de la matrice A dans un espace de dimension k s'appelle l'espace LSI de A.

Afin de montrer comment LSI est capable d'améliorer les résultats d'une recherche par l'extraction de la sémantique interne des données, Deerwester et al. proposent dans [26] un exemple résumé dans la figure 2.2.

Titles:
c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

Terms	**Documents**								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

FIGURE 2.2 – Un exemple d'un corpus simple constitué de 9 titres de documents techniques et la matrice associée [26].

L'exemple représente 9 titres de documents (c_i, $i \in \{1,..5\}$ et m_j, $j \in \{1,..4\}$) groupés en deux catégories : "interaction homme machine" (les c) et "graphes" (les m). Ces titres sont représentés par la matrice $A_{(12,9)}$. Pour cet exemple, Deerwester et al. choisissent avec attention les termes et les documents pour que SVD produise des résultats satisfaisants sur 2 dimensions seulement, soit A_2. Après la décomposition de la matrice, les documents et les termes sont représentés géométriquement dans un espace à deux dimensions. La figure 2.3 visualise la répartition des termes

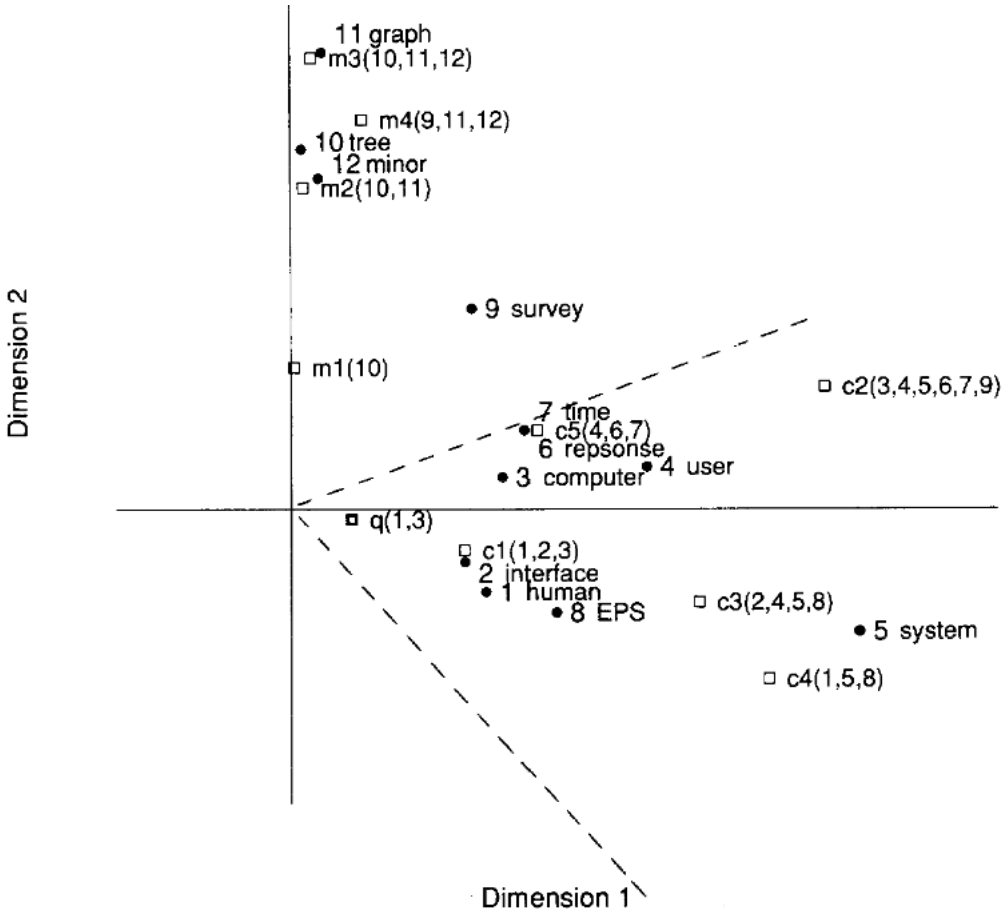


FIGURE 2.3 – Représentation géométrique des documents et termes dans un espace de 2 dimensions [26].

(représentés sous forme de ronds noirs) et les documents (représentés sous forme de rectangles vides avec entre parenthèses les termes associés) dans l'espace. Le but est de trouver les documents qui sont en relation avec le titre requête "Computer Human interaction". Si nous ne tenons compte que des termes, les documents retournés seront seulement ceux qui contiennent au moins une occurrence en commun avec la requête, soit donc c_1, c_2, c_4. Dans ce cas, c_3 et c_5 seront ignorés même s'ils appartiennent au même domaine car ils n'ont aucun terme en commun avec la requête.

Si nous considérons maintenant les résultats de LSI, dans la figure 2.3, la requête est représentée par $q_{(1,3)}$. Elle est bien placée au centroide des deux termes "human", "computer". Si nous regardons maintenant les deux documents c_3 et c_5, on remarque qu'ils sont proches du point $q_{(1,3)}$ ce qui n'est pas le cas des documents m_1, m_2, m_3. En utilisant LSI nous arrivons donc à associer c_3 et c_5 à $q_{(1,3)}$ bien qu'ils ne présentent aucun mot en commun. Les relations entre les documents sont généralement basées sur des associations complexes et indirectes entre eux et les termes. La puissance de LSI réside dans son utilisation d'une structure d'ordre supérieure dans la matrice représentative des données qui permet de trouver la sémantique d'un mot tout seul.

LSI a été conçue principalement pour résoudre des problèmes de recherche d'information, mais elle est aussi utilisée dans des problèmes de classification, notamment la classification des messages courts comme nous le verrons plus loin dans ce document avec les travaux de Zelikoviz [88].

2.3.1.2 Latent Semantic Analysis (PLSA) ou Probabilistic Latent Semantic Indexing (PLSI)

Cette méthode est proposée par Hofmann dans [35]. Il s'agit d'une amélioration de LSI qui se base sur un modèle statistique appelé : modèle d'aspects [36]. Ce modèle se base sur des variables latentes qui sont associées à différentes observations. La probabilité de joindre un terme W à un document D peut être alors définie selon la formule 2.3 et la figure 2.4-a :

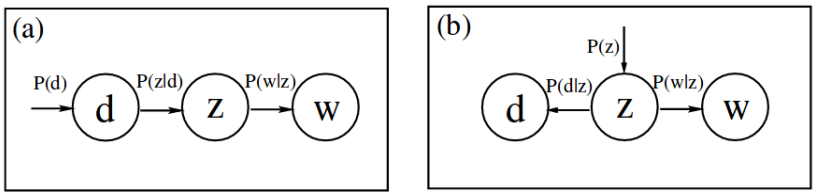


FIGURE 2.4 – Paramétrisation asymétrique et symétrique du modèle d'aspects [35].

$$P(d,w) = P(d)P(w|d) \Rightarrow P(w|d) = \sum_{z \in Z} P(z)P(w|z)P(z|d) \tag{2.3}$$

Le modèle d'aspects part de l'hypothèse que D et W sont indépendants de l'état de la variable latente $z \in Z = \{z_1, z_2, ..., z_k\}$ et comme la cardinalité de z est plus petite que le nombre de documents/mots, Hofmann affirme que le modèle peut être paramétré symétriquement selon la formule 2.4 et la figure 2.4-b :

$$P(d,w) = \sum_{z \in Z} P(z)P(d|z)P(w|z) \tag{2.4}$$

Pour estimer la probabilité de la variable z, Hofmann a utilisé l'algorithme Expectation Maximisation ou EM [27].

Afin de bien comprendre la différence entre LSI et PLSI, Hofmann réécrit le modèle d'aspects (paramétrisation symétrique) sous forme de matrices comme défini dans 2.1. Dans ce cas :

— $U = (P(d_i|z_k))_{i,k}$,
— $D = Diag(P(z_k))_k$
— et $V_t = (P(w_j|z_k))_{j,k}$.

De cette façon PLSI est écrit aussi suivant la formule $U \times D \times V^T$.
Le facteur k dans PLSI représente un mélange de décompositions de la matrice initiale en plusieurs modèles dérivés du modèle d'aspects, alors que LSI se base sur une seule décomposition linéaire (SVD). Le mélange des modèles se substitue à la décomposition singulière des valeurs. Autrement dit, la différence majeure entre PLSI et LSI se trouve principalement dans la fonction objective qui détermine la décomposition optimale de la matrice représentative des données. En effet, LSI utilise la norme $L2$ ou Forbenius qui correspond à une hypothèse implicite d'une gaussienne additive alors que PLSI se base sur un échantillonnage multinomial probabiliste qui vise à maximiser explicitement la capacité de la prédiction du modèle en minimisant l'entropie croisée ou la divergence Kullback-Leiber [47]. Cela signifie que PLSI se base sur une distribution de probabilité bien définie qui est une distribution multinomiale des mots du corpus, alors qu'on ne connaît pas la distribution de probabilités dans LSI.
L'approche probabiliste de PLSI présente aussi un avantage par rapport à LSI grâce à la théorie statistique du modèle de sélection et de contrôle de complexité pour déterminer le nombre optimal de la dimension de l'espace latent (k) alors que k dans LSI est choisi aléatoirement. Néanmoins le temps de calcul de PLSI est plus important que celui de LSI à cause de l'algorithme EM qui est un algorithme itératif.

2.3.1.3 Latent Dirichlet Allocation (LDA)

Cette technique du "topic model" a été proposée par Blei en 2003 dans [13]. LDA est un modèle probabiliste génératif d'une collection de données discrètes telles que par exemple les textes. Il s'agit aussi d'un modèle bayésien hiérarchique construit sur trois niveaux : corpus, document et mot. La figure 2.5 illustre ces trois niveaux. LDA considère qu'un document est un ensemble de thèmes cachés selon une distribution particulière de mots. L'idée générale de LDA est d'extraire, par exemple à partir d'un ensemble de documents textuels, un ensemble de thèmes où chaque thème est un ensemble de mots qui sont sémantiquement proches. Un mot est associé à un thème par un poids qui représente l'importance du mot dans le thème. Ainsi, un mot peu appartenir à plusieurs thèmes à la fois mais avec des poids différents. Afin de comprendre l'algorithme utilisé par LDA pour générer les thèmes, Blei dans [13] donne les définitions suivantes :

— un *mot* w : c'est l'unité basique. C'est l'indice d'un mot dans un vocabulaire noté $\{1, ..., V\}$. On peut considérer que w est un vecteur de taille V où tous les éléments sont mis à 0 sauf le mot en question ;
— un *document* : représente une séquence de N mots noté

$\boldsymbol{w} = (w_1, w_2, ...w_N)$;

— un *corpus* : représente une collection de M documents noté $D = \{d_1, d_2, ..., d_M\}$

L'algorithme de génération peut être résumé dans l'algorithme 1.

Algorithme 1 : Latent Dirichlet Allocation (LDA)

Entrée: Ensemble de documents
Sortie: Ensemble de thèmes
Pour chaque document d dans le corpus D :
choisir $N \sim Poisson(\varepsilon)$
choisir $\theta \sim Dir(\alpha)$
Pour N mots w_n :
choisir un thème $z_n \sim Multinomial(\theta)$
choisir un mot w_n de $p(w_n|z_n, \beta)$, une probabilité multinomiale conditionnée sur le thème z_n

Comme on peut voir dans la figure 2.5, les paramètres α et β sont des paramètres du niveau corpus : ces paramètres sont échantillonnés lors du processus de génération du corpus. La variable θ est une variable de niveau document échantillonnée pour chaque document et finalement les deux variables z_{dn} et w_{dn} sont deux variables de niveau mot, elles sont échantillonnées pour chaque mot de chaque document. Même si LDA suit la loi de Dirichlet dans son processus génératif, cette technique est bien distinguée du modèle du regroupement du Dirichlet-multinomial classique. En effet ce dernier est un modèle construit sur deux niveaux seulement et dans lequel un document ne peut être associé qu'à un seul thème, alors que LDA est constituée sur trois niveaux et elle considère qu'un document peut être associé à plusieurs thèmes à la fois.
Afin d'estimer les différents paramètres du modèle tels que par exemple α et β, Blei dans son implémentation de LDA a adopté l'approche bayésienne empirique présentée dans [54].

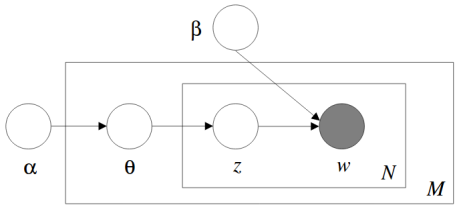


FIGURE 2.5 – Représentation graphique du modèle de LDA [13].

Blei, dans son article [13], a montré les avantages de sa méthode par rapport à PLSI [36]. En effet cette dernière a deux limites : PLSI n'est pas un vrai modèle génératif bien qu'il considère qu'un document contient plusieurs thèmes à la fois avec différents poids. Si nous reprenons son principe, la probabilité qu'un terme soit associé à un document est définie par l'équation 2.4. $P(z|d)$ représente l'ensemble

des poids des thèmes dans un document d, mais il est important de noter que le modèle n'est capable de générer que les thèmes contenus dans les documents d'apprentissage. PLSI n'est donc pas capable de trouver la probabilité d'un document non vu auparavant.
La distribution multinomiale de données utilisée par PLSI est indexée par les documents d'apprentissage. Le nombre de paramètres à estimer dans le modèle augmente linéairement alors avec l'augmentation des données d'apprentissage, ce qui mène à un vrai problème de sur-apprentissage. Ceci constitue une deuxième limite du modèle PLSI.
LDA résout les deux problèmes mentionnés ci-dessus en offrant un vrai modèle génératif qui associe plusieurs thèmes à un document. Blei dans [13] a montré que son modèle ne souffre pas de sur-apprentissage en se basant sur la mesure de perplexité [1].

LDA fait partie des algorithmes d'apprentissage non supervisé où le nombre de thèmes n'est pas connu en avance. Ce nombre reste un paramètre qui doit être défini en avance par l'utilisateur ou empiriquement suivant le type et la quantité des données utilisées. Arun et al. proposent dans [6] une mesure pour identifier le nombre optimal de thèmes généré par LDA par corpus.
LDA étant un modèle génératif probabiliste, il peut être vu comme une méthode de factorisation non négative de matrices où la matrice représentative des données peut être décomposée en deux : une qui contient la distribution des mots dans les thèmes(M_1) et une qui contient la distribution des thèmes dans les données(M_2). En variant le nombre de thèmes, ces deux matrices changent et le but est de comparer les différentes matrices générées afin de trouver le nombre optimal de thèmes. Arun et al. définissent alors une mesure qui compare des propriétés similaires des matrices générées. Cette mesure se base sur la décomposition de la matrice M_1 en valeurs singulières (SVD).

$$M_1 = U \times \Sigma \times V' \tag{2.5}$$

$\sum$ est une matrice diagonale qui représente les valeurs singulières notées par $\sigma_i, i = 1, .., m$, m étant le nombre de thèmes. La distribution de ces valeurs dans M_1 représente la distribution de la variance dans les thèmes. Comme le nombre optimal de thèmes est obtenu quand les mots du vocabulaire sont bien distribués séparément dans les différents thèmes, c'est à dire que les thèmes sont constitués de peu de mots communs, Arun et al. ont démontré qu'une telle séparation est obtenue quand σ_i est égal à la norme L_2 du vecteur de la i^{eme} ligne de la matrice $M_1, \forall i = 1, ..., m$. En se basant sur cette proposition Arun et al. augmentent le nombre de thèmes jusqu'à ce que la distribution des valeurs singulières soit proche de la distribution de la norme L_2 de la ligne.
La mesure finale proposée peut être résumée dans la formule suivante :

$$ProposedMeasure(M_1, M_2) = KL(C_{M_1}||C_{M_2}) + KL(C_{M_2}||C_{M_1}) \tag{2.6}$$

où :
KL est la divergence de Kullback-Leibler [47] ;

1. https ://en.wikipedia.org/wiki/Perplexity

C_{M_1} est la distribution des valeurs singulières de M_1 ;
C_{M_2} est la distribution obtenue par la normalisation de $l \times M_1$ (l est le vecteur qui contient la taille de chaque document dans la matrice M_2).
Les différentes observations que Arun et al. ont fait sur des benchmarks réels ont montré que le nombre optimal de thèmes est obtenu quand la KL divergence est la plus basse.

2.3.2 Enrichissement des messages courts

Pour améliorer la classification des messages courts plusieurs recherches se sont concentrées sur l'amélioration de la qualité des messages avant de construire les modèles de classification, s'attaquant ainsi au problème de faible densité. Pour cela une première communauté de chercheurs s'est dirigée vers l'enrichissement des textes originaux par de l'information apportée de l'extérieur. Généralement, cette information externe est sémantiquement proche des textes initiaux.
Plusieurs sources d'information ont été utilisées dans la littérature pour enrichir les textes comme les ontologies, les documents collectés à la main ou les dictionnaires.

Enrichissement basé sur la sémantique apportée par les ontologies
Les ontologies [60], habituellement constituées de trois parties (concepts, relations et attributs), ont été utilisées pour améliorer la classification des messages courts. Le principal intérêt de cette utilisation est l'apport de la connaissance externe qui joue le rôle d'un pont qui clôt les trous sémantiques entre les mots des messages. Les différents éléments d'une ontologie peuvent soit enrichir soit remplacer le texte des messages courts.
Plusieurs travaux ont utilisé des ontologies universelles comme WordNet [53] dans [38] et [37] et Mesh [2] dans [90] et [87] pour améliorer le regroupement des textes. Mais selon Hu et Zhang dans [40] l'utilisation de ces deux ontologies pour extraire les liens sémantiques entre les textes révèle deux limites majeures :

— la perte d'information due à la couverture partielle de ces ontologies de tous les domaines ;
— l'introduction du bruit à cause de leur caractère très général et donc de leur non spécificité à des domaines particuliers.

Pour faire face à ces deux limites Hu et Zhang proposent une nouvelle plate-forme décrite dans la figure 2.6. Celle-ci consiste à proposer une nouvelle méthode d'enrichissement de textes par les concepts et les catégories de Wikipédia [3] pour améliorer le regroupement des textes. Le choix de Wikipédia est dû à la grande quantité d'information que regroupe cette source et à sa structure pouvant être utilisée comme ontologie puisqu'elle est constituée d'un ensemble d'articles reliés entre eux et identifiés chacun par un titre.

La première étape de l'approche consiste à lier chaque texte à un ou plusieurs concepts puis à une ou plusieurs catégories. Un texte d_i sera donc représenté par un

2. www.nlm.nih.gov/mesh/meshhome.html
3. fr.wikipedia.org

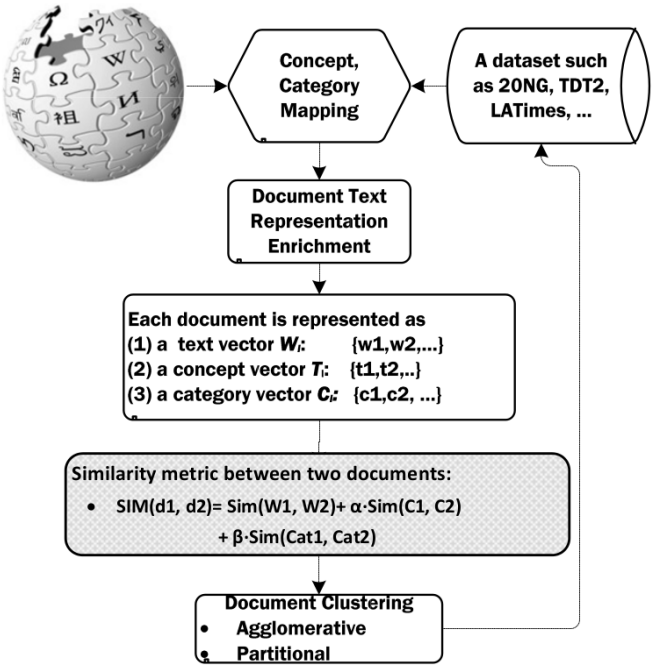


FIGURE 2.6 – Plate-forme générale proposée par Hu et Zhang dans [40] pour améliorer le regroupement de textes en introduisant la sémantique externe par Wikipédia.

vecteur W_i qui est l'ensemble de mots le constituant, un vecteur de concepts C_i et un vecteur de catégories Cat_i. La similarité entre deux textes d_1 et d_2 (SIM) sera calculée ensuite en tenant compte des concepts et catégories de chaque texte selon cette formule [4]

$$SIM(d_1, d_2) = Sim(W_1, W_2) + \alpha Sim(C_1, C_2) + \beta Sim(Cat_1, Cat_2) \tag{2.7}$$

En utilisant cette similarité les textes vont être regroupés selon une approche ascendante et une approche de partitionnement [92].

Pour assurer la liaison entre les textes, les concepts et les catégories, Wikipédia est représentée sous forme d'une matrice où les colonnes sont les catégories, les lignes sont les concepts et les valeurs sont 1 si le concept appartient à la catégorie et 0 sinon. La correspondance entre textes et concepts est faite par deux méthodes qui sont "la correspondance exacte" et "la correspondance reliée". Le résultat de ces deux méthodes est une autre matrice dont les lignes sont les textes, les colonnes sont les concepts et les valeurs sont le poids de chaque concept dans chaque texte. En utilisant ces deux matrices on peut déduire la dernière matrice textes-catégories. La figure 2.7 illustre ces trois matrices.

4. Sim : Similarité

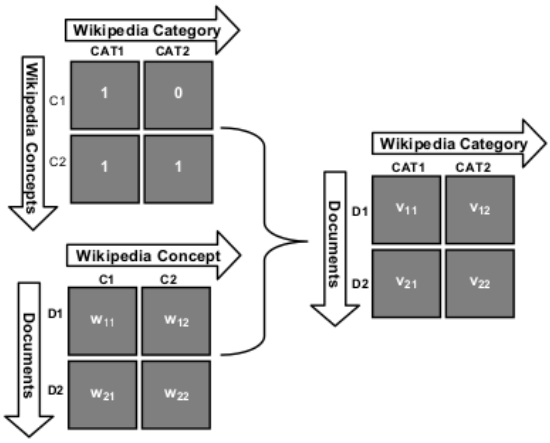


FIGURE 2.7 – Illustration de la correspondance entre les textes, les concepts et les catégories de Wikipédia dans [40].

2.3.2.1 Correspondance exacte

La première étape dans cette méthode consiste à construire l'ensemble des concepts de Wikipédia. Un concept est défini comme le titre d'un article. Plusieurs autres pages pouvant parler du même sujet, les concepts de ces pages seront redirigés vers ce concept principal. Il s'agit de construire un dictionnaire "concept principal, concepts redirigés". Le poids de chaque concept dans chaque texte est calculé suivant le nombre d'apparition de ce concept ainsi que les concepts redirigés dans ce texte, puis on applique la méthode TF.IDF [64] (Fréquence du Terme, Fréquence Inverse de Document). Cette mesure permet d'évaluer l'importance d'un mot dans un texte, relativement à l'ensemble total des textes. Le poids augmente proportionnellement au nombre d'occurrences du mot dans le texte. Il varie également en fonction de la fréquence du mot dans tous les textes. Les concepts qui ont un poids dépassant un certain seuil défini par l'utilisateur sont retenus pour le texte en question.

2.3.2.2 Correspondance reliée

La méthode de correspondance exacte est une méthode rigide puisqu'un concept ne peut être associé à un texte que si, et seulement si, tous les termes du concept se trouvent dans le texte. Pour contourner ce problème, Hu et Zhang proposent une nouvelle méthode de correspondance plus souple où la liaison d'un texte à un concept se base sur l'apparition d'un terme commun. Il s'agit donc de calculer le nombre de fois où chaque terme apparaît dans un concept puis de calculer le nombre de fois où chaque terme apparaît dans un texte. La similarité d'un concept k à un texte d_j est calculée par la formule suivante :

$$r_k^{d_j} = \sum_{w_i \in d_j} TF.IDF_{w_i}^{d_j} \times TF.IDF_{w_i}^{c_k} \tag{2.8}$$

Il s'agit de la somme des produits des poids de chaque terme du texte dans le texte d_j et dans le concept k. Puis les M concepts qui ont les similarités les plus grandes seront associés au texte d_j. Le poids est calculé à l'aide de TF.IDF.

Comparée à la méthode classique (BOW), cette méthode [40] a amélioré la qualité de regroupement de textes sur des bechmarcks utilisés dans la littérature comme TDT2, LA Times et 20-newsgroups.

Dans le même contexte, une autre approche plus spécifique aux messages courts est proposée par Hu et Sun dans [39]. Elle se base aussi sur les ontologies pour résoudre le problème de faible densité des messages courts. Hu et Sun utilisent dans cette nouvelle approche Wikipédia et Wordnet pour enrichir les textes. Avant d'introduire la connaissance externe, les auteurs s'intéressent à la sémantique interne des textes. Il s'agit dans un premier temps de trouver les phrases informatives. Cette phase utilise les outils de traitement automatique de la langue (TAL). Il s'agit de décomposer le message en segments (phase segment) en utilisant la ponctuation, puis en phrases (phase phrase) en utilisant la partie du discours ("part of speach") : décomposer le segment en phrases suivant leur nature grammaticale. Une fois les phrases obtenues, les auteurs identifient les plus informatives en cherchant celles qui sont significatives dans le message et en supprimant celles qui sont redondantes. Cette étape est réalisée par le calcul de similarité sémantique entre les segments et leurs phrases dérivées :
pour chaque phrase p_i du segment $P = \{p_1, ..p_i, .., p_n\}$ on lui applique la mesure de similarité infoScore en tenant compte de toutes les autres phrases du segment. La phrase qui a l'infoScore le plus élevé est la phrase la plus similaire aux autres phrases : c'est donc une phrase redondante. Une fois les phrases informatives sélectionnées, on génère l'information extérieure à partir de deux sources d'enrichissement. La première source est Wikipédia car elle couvre bien tous les domaines et elle est actualisée. La deuxième source est WordNet puisqu'elle est riche en connaissance sémantique et lexicale. La figure 2.8 résume le processus d'enrichissement qui se base sur le moteur de recherche Solr [5]. L'algorithme consiste initialement à identifier la source d'enrichissement adéquate pour chaque phrase informative. Si la phrase contient plus qu'un mot qui n'est pas un mot d'arrêt on utilise alors Wikipédia car la phrase est considérée comme une phrase qui représente un sous thème du message. Des articles de Wikipédia reliés à cette phrase peuvent être trouvés. Dans le cas contraire, WordNet est utilisé pour enrichir le mot avec ses synonymes. Dans le cas où la phrase est enrichie à partir de Wikipédia, elle est traitée sur deux niveaux :

— si elle est dans la phase segment alors une requête **"OU"** est envoyée au moteur de recherche Solr, ce qui signifie que le résultat de la requête doit contenir des articles qui contiennent au moins un mot de la phrase à enrichir ;
— si la phrase appartient à la phase de phrase alors la requête envoyée est **"ET"**. Tous les articles ramenés doivent contenir tous les mots de la phrase à enrichir.

Une fois l'enrichissement fait, Hu et Sun appliquent les techniques de sélection sur

5. lucene.apache.org/solr/

Algorithm 1: GenerateFeatures(S)

input : a set S of *seed phrases*
output: *external features* EF

```
EF ← null
for seed phrase s ∈ S do
    if s.non-stop >1 then
        if s ∈ Segment level then
            s.Query ← SolrSyntax(s, OR)
        else
            s.Query ← SolrSyntax(s, AND)
        WikiPages ← Retrive(s.Query)
        EF ← EF + Analyze(WikiPages)
    else
        EF ← EF + WordNet.Synsets(s)
return EF
```

FIGURE 2.8 – Algorithme de génération de caractéristiques extérieures proposé par Hu et Sun dans [39].

le nouvel espace de caractéristiques composé des caractéristiques du texte d'origine et des caractéristiques externes pour supprimer le bruit.

Hu et Sun ont testé leur méthode sur deux benchmarks : Reuters-21578 [6], après suppression des textes qui sont composés de plus de 50 mots, et Web Dataset qui est extrait de Google tendance [7]. La performance de la méthode a surpassé les méthodes proposées dans la littérature en terme de regroupement de textes courts mais elle reste difficile à mettre en œuvre et non compatible avec les applications temps réel puisque la recherche d'information nécessite beaucoup de temps.

Enrichissement basé sur l'application des topics models sur des documents externes

L'idée d'enrichir les messages courts en appliquant les techniques de "topic models" sur un ensemble de documents externes est introduite par Phan et al. dans [59] en 2008. Le but, toujours le même, est de faire face aux limites de la classification des messages courts que nous venons de mentionner dans la section 2.2 et qui sont principalement la faible densité des messages courts et leur manque de contexte. Au lieu d'utiliser les ontologies, Phan enrichit la base des messages par des thèmes sémantiquement proches. Phan a proposé une nouvelle plate-forme illustrée dans la figure 2.9. Cette plate-forme est composée de 4 étapes majeures :

1. Construction de la base universelle : c'est la source d'enrichissement. Il s'agit de collecter un ensemble de documents externes liés à la base d'apprentissage comme par exemple la collection d'un ensemble d'articles Wikipédia qui sont en relation avec les domaines de la base d'apprentissage. Cette étape est la

6. http ://davidlewis.com/ resouces/testcollections/reuters21578
7. http ://www.google.com/trends

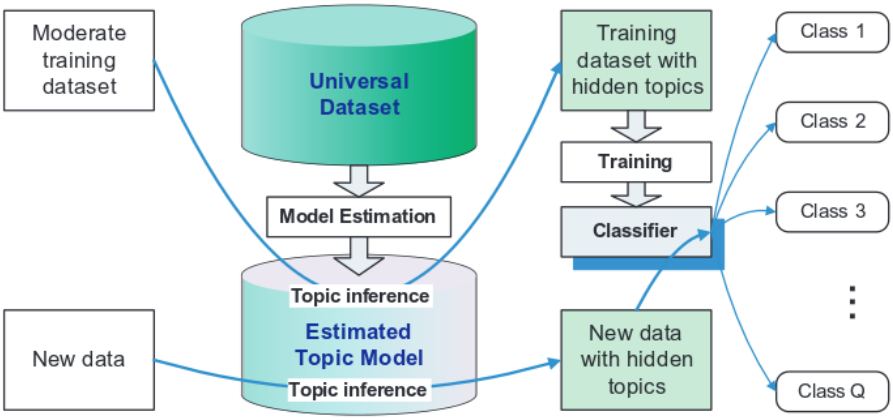


FIGURE 2.9 – Plate-forme générale proposée par Phan pour l'apprentissage des messages courts avec les thèmes cachés [59].

phase la plus cruciale car c'est elle qui va apporter la connaissance externe à l'ensemble de départ. Pour cette raison la base universelle doit premièrement couvrir le plus largement possible tous les domaines de la base d'apprentissage pour assurer un maximum d'enrichissement et deuxièmement être pré-traitée pour ne pas introduire de bruit.

2. Application de LDA [13] sur la base universelle : une fois les documents externes construits et pré-traités, on cherche les thèmes cachés dans la base universelle en utilisant la technique de LDA. Le choix de LDA se justifie par le fait que cette technique a une hypothèse de génération de documents plus complète que les autres techniques.
3. Construction de la base d'apprentissage : un ensemble modéré de messages courts est annoté. Il n'est pas nécessaire d'annoter un grand nombre de messages puisque la base initiale sera enrichie par les thèmes découverts à l'étape précédente.
4. Intégration des thèmes dans les messages : c'est la dernière étape de ce processus. Elle consiste à enrichir les messages par les thèmes en utilisant des règles d'inférence qui se basent sur l'importance des thèmes dans les messages.

Cette méthode a amélioré considérablement la classification de deux bases de messages courts qui sont :

— **"web search snippets"** : un corpus collecté par Phan lui même. La description détaillée de ce corpus sera présentée dans le chapitre 4.
— **"Ohsumed/MEDLINE** : c'est une collection de 233.442 résumés de 270 journaux médicaux qui ont été publiés pendant la période de 1987 à 1991. Le corpus est obtenu après élimination des résumés très courts, des méta-données et des mots d'arrêt.

La plate-forme proposée par Phan permet d'améliorer la classification des messages courts grâce aux points suivants :

— l'augmentation de la densité des données en renforçant la distinction des mots rares du corpus et en clarifiant le sujet précis des messages.
— l'extension de la couverture de la classification en ajoutant de nouveaux termes externes à la base d'apprentissage initiale ;
— la flexibilité de la plate-forme semi-supervisée puisqu'elle construit un classifieur en se basant à la fois sur un ensemble de données labellisées (la base initiale) et sur un ensemble de données non labellisées (les thèmes) avec un avantage par rapport à l'apprentissage semi supervisé traditionnel qui est la non exigence du même format pour les données labellisées et non labellisées ;
— le coût de construction de la base d'apprentissage est faible puisqu'on n'exige pas l'annotation d'un grand nombre de messages.

Bien que Phan a prouvé l'efficacité de sa méthode pour la classification des messages courts, en 2011 Chen et al. dans [21] ont souligné une éventuelle limite de cette méthode. Cette limite se situe dans le processus d'extraction des thèmes à partir des documents universels. En effet, l'extraction des thèmes est faite sur un seul niveau, ce qui peut mener à des thèmes très génériques et à fausser l'algorithme d'inférence de thèmes en associant par exemple deux messages de classes différentes à un seul thème trop général. Pour résoudre cette problématique, Chen et al. proposent une nouvelle méthode qui permet d'associer à chaque message des thèmes plus précis avec un niveau de granularité beaucoup plus fin. La figure 2.10 illustre la méthode proposée par Chen et al.

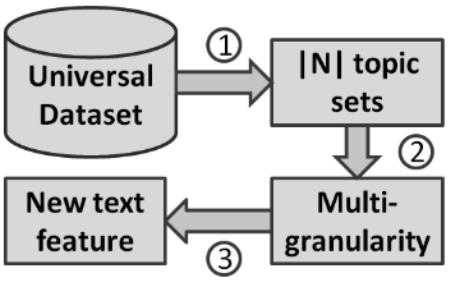


FIGURE 2.10 – Plate-forme de multi-granularité proposée par Chen et al. dans [21].

Cette méthode est composée de trois étapes :

1. générer $|N|$ sous ensembles de thèmes en utilisant LDA : le but de cette étape est de construire un ensemble de sous thèmes de différentes granularités à partir de la base universelle. Il s'agit de choisir empiriquement un ensemble N de nombres. $|N|$ représente le nombre total des sous ensembles de thèmes générés. La taille de chaque sous ensemble est égale à l'élément qui lui correspond dans N.

   On note $T = \{T_1, T_2, ..., T_{|N|}\}$ l'ensemble total des sous ensembles de thèmes générés, et $T_i = \{z_{i1}, z_{i2}, ..., z_{iK_i}\}$ le $i^{ème}$ sous ensemble composé de K_i thèmes et z_{ij} représente un thème.

2. choisir le meilleur sous ensemble parmi les ensembles générés : la qualité des

sous ensembles dépend de leur capacité à offrir des thèmes composés de caractéristiques discriminantes pour la classification. La qualité de l'ensemble dépend aussi de la distance entre ses thèmes qui doivent être suffisamment loin l'un de l'autre pour éviter les problèmes d'ambiguïté. En se basant sur ces deux constatations Chen et al. dans [21] proposent un nouvel algorithme de sélection qui permet de générer un espace de thèmes avec différentes granularités. La première étape consiste à pondérer les sous ensembles de l'ensemble global T pour obtenir un nouvel ensemble $W = \{w(T_1), w(T_2), ..., w(T_{|N|})\}$, $w(T_i)$ étant le poids du thème T_i. L'algorithme de pondération est basé sur l'idée de Relief [46] : le poids de chaque T_i dans W est mis à jour par chaque message de la base d'apprentissage. En effet, pour chaque message x de la base D, on cherche le voisin le plus proche de x et qui appartient à la même classe ($nh(x)$) et le voisin le plus proche de x mais qui appartient à une classe différente ($nm(x)$). Le poids est calculé par la formule suivante :

$$w(T_i) = w(T_i) + d(x_{T_i}, nm(x)_{T_i}) - d(x_{T_i}, nh(x)_{T_i}) \tag{2.9}$$

Une fois l'ensemble W obtenu, on choisit alors le sous ensemble le moins redondant. Cela se fait en calculant la distance entre chaque couple de sous ensembles de thèmes $D(T_i, T_j)$ en utilisant la divergence Kullback-Leiber. Soit :

$$D(T_i, T_j) = D(T_j, T_i) = \sum_{z_i \in T_i, z_j \in T_j} D(p(w|z_i), p(w|z_i))/(K_i \times K_j) \tag{2.10}$$

Soit O l'ensemble des sous ensembles de thèmes choisis. Le score final de chaque sous ensemble dépend de son poids et des sous ensembles déjà choisis. Soit :

$$Score(T_i) = w(T_i) + \sum_{T^{(j)} \in O} Dis(T_i, T^{(j)}) \tag{2.11}$$

3. ajouter les nouvelles caractéristiques au messages et construire le modèle de classification.

Chen a testé sa méthode sur le corpus "web search snippets" construit par Phan [59] et il a montré que sa méthode a surpassé celle de Phan.

2.3.2.3 Enrichissement basé sur la traduction des messages en multi-langues

Un autre type d'enrichissement qui se base sur la puissance des machines de traduction comme Google traduction, Yahoo ou Babel Fish est proposé par Tang et al. en 2012 dans [77].
Le but est d'élargir l'espace des caractéristiques par la connaissance apportée par la traduction des messages courts en d'autres langues afin de résoudre certains problèmes comme la courte taille des messages, les abréviations, la synonymie et la

polysémie. L'idée de traduire un message en plusieurs langues est inspirée des observations suivantes :

1. plusieurs mots qui sont synonymes dans une langue peuvent être traduits en un seul mot dans une autre langue, par exemple : "firm" ou "company" en anglais se transforment en "entreprise" en français. Si nous passons à la langue française, en construisant la matrice représentative des données, ces deux mots ne seront plus considérés comme étant une seule occurrence de deux caractéristiques différentes mais plutôt comme deux occurrences d'une même caractéristique.
2. en traduisant la totalité du message, les machines de traduction conservent l'information contextuelle, ce qui permet de résoudre certains problèmes de polysémie. Par exemple, le mot "saw" en anglais peut avoir le sens d'une "scie" dans la phrase " I cut the wood by the **saw**" et le sens du verbe "voir" dans la phrase "I **saw** my mother". En les traduisant en français cette ambiguïté sera résolue en obtenant "je coupe le bois à la **scie**" et "j'ai **vu** ma mère". Il est possible de tomber dans le problème de polysémie en traduisant un mot d'une langue à une autre. Cependent, sa traduction en plusieurs langues à la fois permet de résoudre ce problème.
3. certaines machines de traduction arrivent à traduire une abréviation dans une langue en un mot complet dans une autre langue. On prend par exemple le mot "Abbr" en anglais qui se transforme en "abréviation" en français.

A partir de ces trois observations Tang et al. [77] proposent une nouvelle plate-forme illustrée dans la figure 2.11

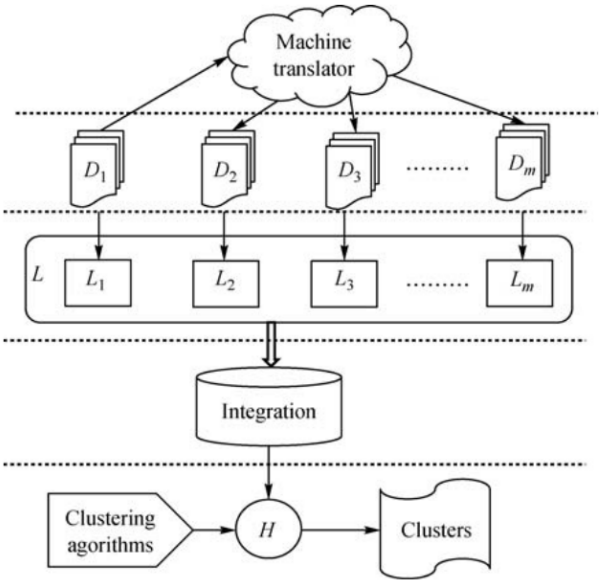


FIGURE 2.11 – Plate-forme générale proposée par Tang et al. pour le regroupement de messages courts en utilisant la connaissance multi-langues [77].

L'approche consiste à traduire les messages courts de la langue d'origine dans un ensemble d'autres langues, puis de construire la matrice représentative de chaque ensemble de messages courts de chaque langue en se basant sur TF.IDF [64]. Une fois que les matrices sont construites, une phase d'intégration de toutes ces matrices en une seule est appliquée. Le processus d'intégration se base sur la réduction du nombre de caractéristiques grâce à la factorisation matricielle.

Phase d'enrichissement

Soit $T = \{t_1, t_2, ..., t_m\}$ un ensemble de m langues ou t_1 est la langue d'origine.
Soit $D_1 = \{d_{(1,1)}, d_{(1,2)}..., d_{(1,n)}\}$ la base de n messages courts de la langue d'origine.
Chaque message $d_{(1,j)}$ va être traduit en $m-1$ langues.
Soit donc $D_i = \{d_{(i,1)}, d_{(i,2)}..., d_{(i,n)}\}$ la i^{eme} base obtenue pour la i^{eme} langue. Soit $L = \{L_1, L_2, ..., L_m\}$ l'ensemble des matrices représentatives des bases $D_1, D_2, ...D_m$.

Phase d'intégration

La phase d'intégration consiste à fusionner l'ensemble L de toutes les matrices en une seule. Les auteurs de l'article ont écarté l'idée d'élargir le vocabulaire initial obtenu par la langue d'origine par tout le vocabulaire des autres langues et ceci pour les raisons suivantes :

1. l'élargissement de l'espace de caractéristiques entraîne un vecteur plus grand, ce qui va diminuer la densité des messages, déjà faible.
2. la traduction n'est pas fiable à 100%, elle peut introduire du bruit.

Pour éviter ces problèmes Tang et al. [77] proposent une solution qui se base sur la technique de factorisation matricielle.

Pour tester la méthode, Tang et al. ont construit deux benchmarks en se basant sur les deux réseaux sociaux : Facebook [8] et Twitter [9]. Il s'agit d'une collection des statuts les plus pertinents des 30 thématiques les plus populaires sur Google tendance. Le tableau 2.1 présente ces deux corpus.

Table 2.1 – Répartition des corpus Facebook et Twitter [77].

corpus	**Facebook**	**Twitter**
nombre de messages	3578	2430
nombre de classes	30	30
taille du vocabulaire	10502	7680
moyenne des mots/message	22.72	17.43

Tang et al. ont utilisé 5 langues. La langue d'origine de leurs corpus est l'anglais et ils les ont traduits en français, allemand, italien et espagnol.
La méthode a permis d'améliorer le regroupement de ces deux corpus de messages

8. www.facebook.com
9. www.twitter.com

par rapport aux méthodes traditionnelles. Tang et al. ont démontré que leur méthode est évolutive linéairement avec l'augmentation du nombre de messages et du nombre de langues. Néanmoins, cette méthode reste difficile à mettre en œuvre puisqu'il faut avoir un bon traducteur et déterminer les langues à utiliser et leur nombre à l'avance.

2.3.3 Réduction des messages courts

Dans cette section nous nous intéressons à une autre méthode d'amélioration de classification de messages courts. Contrairement à l'enrichissement, une communauté de chercheurs s'est dirigée vers la réduction du nombre de caractéristiques, conduisant ainsi a une représentation matricielle plus simple avec moins de caractéristiques, donc plus dense. Le temps de calcul est ainsi plus petit que celui demandé par l'enrichissement.
Parmi les techniques de réduction, on peut citer l'abstraction, la réduction en utilisant les "Topics Models" et la sélection des caractéristiques.

2.3.3.1 Abstraction

D'une façon générale, l'abstraction, consiste à regrouper les caractéristiques similaires dans un même ensemble plus abstrait et à considérer cet ensemble comme une seule unité. Dans le cadre des textes, l'abstraction est le regroupement sémantique des mots en un seul thème. Le thème correspond au mot le plus abstrait. Ainsi, par exemple : ordinateur, clavier, logiciel, informatique seront regroupés ensemble dans le thème informatique.

En combinant l'abstraction avec les k_grammes [50], Silvescu dans [72] a pour but d'améliorer la classification de séquences comme par exemple les séquences de protéines. L'abstraction dans sa méthode a joué le rôle d'un régulateur du nombre de caractéristiques qui explose en utilisant les k_grammes puisqu'il s'agit de construire de nouvelles caractéristiques plus complexes en combinant celles existantes. Les expérimentations sur cette méthode ont montré que le gain apporté par l'utilisation des k_grammes a été conservé en réduisant le nombre de caractéristiques. Cela permet d'obtenir un modèle optimisé et efficace.

L'algorithme général proposé par Silvescu peut être résumé dans les quatre points suivants :

- générer les nouvelles caractéristiques en utilisant les k_grammes ;
- construire la nouvelle représentation matricielle de données en utilisant les nouvelles caractéristiques ;
- construire l'ensemble réduit de caractéristiques par l'abstraction ;
- régénérer la représentation finale de données en utilisant le résultat de l'abstraction.

La première étape de l'algorithme est la super structuration ou la génération des k_grammes. Elle consiste à générer un ensemble $S = \{s_1, s_2, .., s_n\}$ de sous-chaînes uniques et complexes de taille inférieure ou égale à k à partir d'une base d'apprentissage $D = (x_l, y_l)_{l=1,..,N}$. Soit n la cardinalité de S, $n = |S| <= |X|^k$ où

X est l'alphabet de D. Le modèle construit à partir de la nouvelle base D_s a permis d'améliorer la classification. Cependant, pour le simplifier, l'abstraction est utilisée pour construire une nouvelle base D_{s+A} et réduire le nombre de caractéristiques en m caractéristiques seulement ($m < n$). Il s'agit donc de créer un ensemble $A = \{a_1 : S_1, .., a_n : S_n\}$ où a_i est l'abstraction du sous ensemble S_i. La figure 2.12 résume l'algorithme d'abstraction.

Algorithm 2 `Construct-Abstractions`

Input: $\mathcal{D}_\mathcal{S} = ([s_1 : \#s_1^l], \cdots, [s_n : \#s_n^l], y_l)$, m = number of abstractions to construct
Output: A set of abstractions $\mathcal{A}$, such that $|\mathcal{A}| = m$
Initialize $\mathcal{A} = \{a_1 :\{s_1\}, \cdots, a_n :\{s_n\}\}$
for $u = n + 1$ **to** $2n - m$ **do**
  $(i_{min}, j_{min}) = \arg\min_{i,j} dist(a_i, a_j)$
  $a_u = a_{i_{min}} \cup a_{j_{min}}$
  $\mathcal{A} = \mathcal{A} \backslash \{a_{i_{min}}, a_{j_{min}}\} \cup \{a_u\}$
end for

FIGURE 2.12 – Algorithme de l'abstraction proposé par Silvescu [72].

Initialement chaque sous-ensemble S_i de l'abstraction a_i dans l'ensemble A ne contient qu'un seul élément (une seule caractéristique). A chaque itération l'algorithme cherche les deux abstractions les plus similaires (a_i,a_j), les fusionne dans un seul groupe a_u, supprime a_i et a_j de l'ensemble A et y ajoute a_u. L'algorithme s'arrête quand on obtient m abstractions représentant le nouvel espace de caractéristiques, c'est-à-dire après $n - m$ itérations. Le calcul de similarité entre deux abstractions est basé sur l'observation suivante :
La similarité entre deux éléments se produit avec des contextes similaires. Deux abstractions sont donc similaires quand leurs contextes respectifs le sont aussi. La classe contexte d'une abstraction est l'agrégation des classes contextes de ses éléments. La classe contexte d'une caractéristique s_i est la probabilité conditionnelle de la classe variable Y donnée par s_i. Soit :

$$CContext_D(s_i) := [p(Y|s_i), \#s_i] = \left[\left[\frac{\#[s_i, yi]}{\sum_{y_i \in Y} \#[s_i, y_i]}\right]_{y_i \in Y}, \sum_{y_i \in Y} \#[s_i, y_i]\right] \quad (2.12)$$

$\#[s_i, yi]$ est le nombre d'apparitions de la caractéristique s_i dans la classe y_i dans toute la base d'apprentissage D.

Le contexte de l'abstraction $a_i = \{s_{i_1}, ..., s_{i_q}\}$ sera donc :

$$CContext(a_i) = CContext(\{s_1, s_2, .., s_q\}) = p(y_i|a_i) = [\sum_{r=1}^{q} \pi_r p(Y|s_{ir}), \sum_{i=1}^{q} \#s_{ir}] \quad (2.13)$$

avec : $\pi_r = \frac{\#s_{ir}}{\sum_{i=1}^{q} \#s_{ir}}$.

Une fois les classes contextes des abstractions définies, on calcule ensuite la distance entre les contextes. Comme le but est de réduire l'ensemble A en fusionnant a_i et a_j en a_u, Silvescu [72] a utilisé le principe de l'information mutuelle entre A et la classe variable Y, $I(A, Y)$. La distance entre deux abstractions a_i et a_j est :

$$dist(a_i, a_j) = \delta I(\{a_i, a_j\}, a_u) = (p(a_i)+p(a_j)) \times JS(CContext_D(a_i), CContext_D(a_j)) \tag{2.14}$$

où $JS(CContext_D(a_i), CContext_D(a_j))$ est la divergence pondérée de Jensen Shannon avec les poids π_i et π_j. $\pi_i = \frac{p(a_i)}{p(a_i)+p(a_j)}$, $\pi_j = \frac{p(a_j)}{p(a_j)+p(a_i)}$.

Le principe de l'abstraction est bien mis en œuvre avec Silvescu dans son article [72]. Les résultats des expérimentations proposées sur les deux benchmarks de localisation des sous cellules de protéines **plant** et **non-plant** [10] ont montré que la réduction du nombre de caractéristiques par l'abstraction permet de créer un modèle moins complexe tout en gardant l'efficacité de la classification des séquences de protéines.

D'une façon similaire, l'abstraction a montré son efficacité dans la classification des messages courts. Caragea par exemple dans [18] a appliqué la méthode d'abstraction proposée par Silvescu [72] sur le jeu de données "Ushahidi" [11]. Ce jeu de données est composé de 2116 messages classés manuellement dans 10 catégories. Ces derniers représentent des messages d'appel d'urgence (généralement des Tweets et des SMS) qui ont été émis par les habitants de Haïti lors d'un séisme. Afin de répondre d'une façon rapide à ces appels d'urgences, les messages reçus doivent être classés automatiquement en différentes catégories pour les router au service convenable. Pour cette raison, la méthode de classification utilisée doit être la plus performante possible. Caragea dans [18] a fait une étude comparative sur plusieurs méthodes de réduction de caractéristiques des messages courts en les appliquant sur ce corpus.

Les résultats obtenus sur le corpus d'Ushahidi ont montré que l'abstraction des messages a été meilleure que les autres méthodes de réductions de caractéristiques ainsi que la représentation traditionnelle des données (Bag Of Words).

Un autre travail qui utilise l'abstraction pour la classification des messages courts est proposé par Yang dans [85]. Il s'agit d'une méthode qui réduit le nombre de caractéristiques des messages courts en faisant une correspondance entre les mots des messages et des thèmes plus généraux. La construction des thèmes est faite en suivant le principe proposé par Phan dans [59]. Il s'agit de collecter un ensemble de documents de Wikipédia sémantiquement proches de l'ensemble des messages et de construire les thèmes en appliquant LDA sur cet ensemble. Ainsi, l'espace des caractéristiques sera réduit au nombre des thèmes extraits. Le poids de chaque thème η *a priori* est le nombre d'apparitions du thème dans le message, c'est le nombre d'apparitions des mots qui sont à la fois dans le thème et dans le message. Pour améliorer cette représentation, Yang a pensé à augmenter les poids des thèmes les plus importants en calculant la discrimination des mots des messages avant de

10. http ://www.cbs.dtu.dk/services/TargetP/datasets/datasets.php
11. http ://haiti.ushahidi.com/

faire correspondre les mots et les thèmes. L'importance des mots est obtenue en appliquant une version améliorée de la mesure Entropie croisée attendue ("cross entropy expected"). Cette mesure considère à la fois la fréquence du mot et sa relation avec les catégories des messages. Elle est définie comme suit :

$$f(w) = p(w) \sum_i p(C_i|w) log \frac{p(C_i|w)}{p(C_i)} \tag{2.15}$$

L'importance d'un mot suivant l'entropie croisée attendue classique est la même pour toutes les catégories, alors que dans la plupart des cas un mot ne peut pas être important dans toutes les catégories. En se basant sur cette observation Yang propose une version de calcul d'entropie croisée d'un mot dans une catégorie. La définition de cette nouvelle entropie croisée est :

$$f(w, C_i) = p(w)p(C_i|w) log \frac{p(C_i|w)}{p(C_i)} \tag{2.16}$$

Une autre constatation que Yang a faite est qu'un mot distinct est un mot qui a une grande relation avec une catégorie et une faible relation avec toutes les autres catégories. La mesure finale appelé **M-ECE** est :

$$F(w, C_i) = f(w, C_i) - \sum_{j \neq i} f(w, C_j) \tag{2.17}$$

Au moment où la correspondance mot - thème est faite, si le mot courant est très corrélé à une catégorie, nous supposons que le thème correspondant est lui même très corrélé avec cette catégorie. Nous augmentons donc son poids en ajoutant le poids du mot calculé par M-ECE. De cette façon la représentation matricielle est plus consistante.

Les expérimentations menées par Yang sur sa nouvelle méthode intégrant la réduction des caractéristiques dans le processus de classification de messages courts ont montré des résultats très prometteurs puisqu'il y a une amélioration nette par rapport aux algorithmes classiques. Cependant, il semble que sa méthode est plutôt efficace sur les benchmarcks équiprobables. En effet ses tests ont été réalisés sur 5 catégories équiprobables parmi les 8 catégories du benchmark "search snippets" et parmi les 23 catégories du benchmark "Ohsumed".

2.3.3.2 Réduction par l'utilisation des techniques du topic models

Une autre approche de réduction du nombre de caractéristiques des messages courts est proposée par Zelikovitz dans [88]. Comme dans le cas de [59] et [21], Zelikovitz vise à améliorer la classification en limitant quelques problèmes reliés à la nature des données textuelles. En effet la représentation traditionnelle des données (BOW) qui se base sur le calcul du nombre d'occurrences des mots dans le texte n'est pas très efficace puisqu'elle ne permet pas de considérer la sémantique des textes, comme évoqué dans [40]. Ainsi, avec cette représentation, deux textes

qui n'ont aucun mot en commun sont considérés comme différents même s'ils sont sémantiquement proches, alors que deux textes qui partagent quelques mots en commun sont considérés comme textes similaires bien qu'ils peuvent ne pas l'être à cause de la polysémie (ambiguïté). Pour résoudre ces problèmes Zelikovitz propose dans sa nouvelle méthode d'apprentissage de transformer la matrice initiale en une autre plus réduite qui capte mieux les liens sémantiques entre les termes. Mais avant la réduction, Zelikovitz a opté pour une méthode de calcul de poids qui augmente l'importance des mots distingués pour construire sa matrice traditionnelle $M_{(t,d)}$. Pour chaque terme (mot) t on associe 0 s'il n'existe pas dans le texte d, et un poids calculé à base de nombre d'occurrences locales et globales de ce terme. Le poids est calculé de la façon suivante :

$$poids(t) = pLocal(t) * pGlobal(t) \tag{2.18}$$

$$pLocal(t) = log(occ(t)) \tag{2.19}$$

où $occ()$ est le nombre d'occurrences de t dans d.

$$pGlobal(t) = Entropie(t, c) \tag{2.20}$$

Le calcul de l'entropie se base sur le nombre d'occurrences du terme t dans la totalité du corpus c.

$$Entropie(t, c) = 1 - \sum_d \frac{p_{td} \times log(p_{td})}{log(n)} \tag{2.21}$$

où n est le nombre de textes et p_{td} est le quotient du nombre d'occurrences de t dans d par le nombre d'occurrences de t dans tout le corpus c.

Pour la réduction de la matrice $M_{(t,d)}$, Zelikovitz utilise la technique du topic model LSI afin de capter les liens sémantiques entre les termes. Comme défini dans la section 2.3.1, LSI est une technique qui se base sur l'hypothèse de l'existence d'une structure sémantique cachée dans un ensemble de textes. Cette structure décrit la relation entre les termes et les textes. Nous rappelons que LSI utilise le SVD (Singular Value Decomposition) pour décomposer la matrice initiale en trois sous matrices :

$$M(t, d) = T \times S \times D^T$$

T et D sont les deux matrices gauche et droite et S une matrice diagonale. Les valeurs de la diagonale de S sont ordonnées suivant leurs ampleurs. La matrice peut être simplifiée en mettant les k valeurs les plus petites à 0. Les colonnes dans T et D correspondantes aux k valeurs mises à 0 sont supprimées. Une nouvelle matrice $\widehat{X}$ est générée à partir du produit de ces trois matrices après la suppression des k valeurs. $\widehat{X}$ représente une approximation de la matrice representative des messages. Elle représente un ensemble de facteurs qui sont considérés comme des relations entre les différents mots.

Dans le même contexte mais d'une façon différente, parmi les méthodes utilisées par Caragea dans [18] pour son étude comparative entre les différentes représentations des messages courts, une d'entre elles consiste à réduire l'espace des caractéristiques en utilisant LDA [13]. En effet, son nouvel espace de caractéristiques est

constitué de l'union de l'ensemble des mots qui ont un poids supérieur à un seuil dans chaque thème.

2.3.3.3 Sélection de caractéristiques

Une autre technique de réduction du nombre de caractéristiques qui a été utilisée dans les travaux sur les messages courts et dans la classification des textes d'une façon générale est la sélection de caractéristiques. Cette méthode vise à réduire la taille de la représentation matricielle des messages en choisissant les caractéristiques les plus importantes et en supprimant celles redondantes et peu discriminantes. Plusieurs algorithmes de sélection de caractéristiques sont proposés dans la littérature. Yang et Pedersen résument quelques uns dans [86] :

gain d'information :
le gain d'information est une mesure utilisée dans la sélection de caractéristiques très employée dans l'apprentissage automatique. Elle mesure la pureté de chaque caractéristique en calculant la quantité de l'information obtenue pour chaque catégorie pour cette caractéristique. Soit $\{c_i\}_{i=1}^{m}$ l'ensemble des catégories.
Le gain d'information d'une caractéristique t est calculée par la formule suivante :

$$\begin{aligned} G(t) = &-\sum_{i=1}^{m} P_r(c_i) log Pr(c_i) \\ &+P_r(t)\sum_{i=1}^{m} P_r(c_i|t) log P_r(c_i|t) \\ &+P_r(t)\sum_{i=1}^{m} P_r(c_i|t) log P_r(c_i|t) \end{aligned} \tag{2.22}$$

information mutuelle :
l'information mutuelle entre deux variables x et y est le degré de dépendance entre elles au sens probabiliste. Dans notre cas, il s'agit de mesurer la dépendance entre les caractéristiques et les catégories. D'une façon plus générale c'est la mesure de la réduction de l'incertitude de la catégorie si nous utilisons la caractéristique en jeu.
Soient :

- t une caractéristique,
- c une catégorie,
- A le nombre de fois où t et c sont apparues ensembles,
- B le nombre de fois où t est apparue sans c,
- C le nombre de fois où c est apparue sans t,
- et N le nombre total de textes.

L'information mutuelle entre t et c est définie par :

$$I(t,c) = log\frac{P_r(t \wedge c)}{P_r(t) \times P_r(c)} \tag{2.23}$$

et elle peut être estimée par :

$$I(t,c) \approx log\frac{A \times N}{(A+C) \times (A+B)} \tag{2.24}$$

l'information mutuelle entre t et c est égale à 0 si t et c sont indépendantes. Pour mesurer le score d'une caractéristique par rapport à tous les textes Yang et Jan proposent deux façons de faire résumées par les deux formules suivantes :

$$I_{avg(t)} = \sum_{i=1}^{m} P_r(c_i)I(t,c_i) \tag{2.25}$$

et

$$I_{max(t)} = max_{i=1}^{m}\{I(t,c_i)\} \tag{2.26}$$

CHI square ou X^2 :
CHI square ou X^2 est une mesure statistique qui permet de calculer l'indépendance entre la caractéristique t et la catégorie c. Si nous gardons les mêmes définitions des variables A, B, C mentionnées dans le paragraphe précédent, et si nous définissons D comme étant le nombre de fois où ni t ni c ne sont apparues, alors

$$X^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)} \tag{2.27}$$

X^2 est une valeur normalisée dont la performance dépend de la qualité de la matrice représentative des données. Une fréquence basse d'une caractéristique appartenant à la matrice peut facilement impacter cette normalisation. X^2 est donc connue pour sa non fiabilité dans le cas de basses fréquences des termes.

Dans le cadre des messages courts, Caragea lors de son étude comparative entre les différents types de représentations matricielles des messages [18], a utilisé l'algorithme de relief introduit par Kira et Rendell dans [43] pour la sélection des caractéristiques.
L'algorithme de relief est résumé dans la figure 2.13. Il est robuste au bruit et il se base sur le calcul de poids des caractéristiques. Pour chaque instance X de la base d'apprentissage, l'algorithme cherche l'instance la plus proche de la même classe : $near-hit$ et l'instance la plus proche de la classe opposée : $near-miss$ en utilisant la distance euclidienne, puis pour chaque triplet d'instances il met à jour le vecteur des poids des caractéristiques en utilisant la formule suivante :

$$W_i = W_i - \text{diff}(x_i, near - hit_i)^2 + \text{diff}(x_i, near - miss_i)^2 \tag{2.28}$$

où :
- W_i est le poids de la $i^{ème}$ caractéristique,
- x_i est la valeur de la $i^{ème}$ caractéristique dans X,
- $near - hit_i$ est la valeur de la $i^{ème}$ caractéristique dans $near - hit$,
- $near - miss_i$ est la valeur de la $i^{ème}$ caractéristique dans $near - miss$

— et $\text{diff}(x_i, y_i) = (x_i - y_i)/nu_i$ où nu_i sert à normaliser la différence dans l'intervalle $[0, 1]$.

Une fois le calcul des poids fait, l'algorithme choisit les caractéristiques ayant un poids supérieur à un seuil fixé par l'utilisateur.

```
Relief(𝒮, m, τ)
    Separate 𝒮 into 𝒮⁺ = {positive instances} and
        𝒮⁻= {negative instances}
    W = (0, 0, . . . , 0)
    For i = 1 to m
        Pick at random an instance X ∈ S
        Pick at random one of the positive instances
            closest to X, Z⁺ ∈ 𝒮⁺
        Pick at random one of the negative instances
            closest to X, Z⁻ ∈ 𝒮⁻
        if (X is a positive instance)
            then    Near-hit = Z⁺; Near-miss = Z⁻
            else    Near-hit = Z⁻; Near-miss = Z⁺
        update-weight(W, X, Near-hit, Near-miss)
    Relevance = (1/m)W
    For i = 1 to p
        if (relevance_i ≥ t)
            then    f_i is a relevant feature
            else    f_i is an irrelevant feature

update-weight(W, X, Near-hit, Near-miss)
    For i = 1 to p
        W_i = W_i - diff(x_i, near-hit_i)² + diff(x_i, near-miss_i)²
```

FIGURE 2.13 – Algorithme de relief proposé par Kira et Redell dans [43].

Sriram et al. dans [74] ont utilisé également la méthode de sélection des caractéristiques pour classer les tweets dans des catégories prédéfinies. Pour résoudre le problème de parcimonie des données, les caractéristiques utilisées dans l'approche de Sriram ne sont pas seulement les mots composant les messages mais il s'agit aussi de sélectionner un sous-ensemble de caractéristiques à partir des méta données reliées aux tweets telles que par exemple l'information sur les auteurs des messages, type de tweets, notion temporelle etc... Les expérimentations faites sur les tweets ont montré que cette approche surpasse celle utilisant la représentation "bag of word" (BOW) traditionnelle. Elles ont aussi montré que la classification basée seulement sur les caractéristiques sélectionnées à partir des méta données des tweets est aussi efficace que quand elles sont combinées avec le BOW.

Une autre méthode de réduction des messages courts utilisant la sélection des caractéristiques est proposée par Sun dans [76]. Cette méthode consiste à sélectionner les mots les plus pertinents de chaque message, puis à ne prendre en considération que ces mots pertinents pour classer le message. Cette sélection est faite par le biais de la clarté, une mesure de similarité entre les documents. Pour savoir si un mot w est pertinent ou pas, on forme une requête qui n'est constituée que par w puis on récupère les N documents les plus proches de ce terme O_w sur une collection de messages courts D. Le score du mot w est calculé alors en fonction de ces N textes

retournés selon la formule suivante :

$$\sum_{w' \in V} P(w'|Q_w) log(\frac{P(w'|Q_w)}{P(w'|D)}) \tag{2.29}$$

où V représente l'ensemble du vocabulaire.
Si les mots composant l'ensemble Q_w sont spécifiques à un thème particulier, ils doivent alors avoir un poids important dans Q_w et un poids faible dans la totalité de la collection, sinon l'ensemble retourné représente juste un ensemble aléatoire de textes et le mot n'est pas pertinent. Combiné avec TF.IDF, la méthode proposée par Sun a permis d'obtenir des scores remarquables sur le corpus "search snippets". Les meilleurs résultats sont obtenus en réduisant le message initial en très peu de mots.

2.4 Apprentissage des messages courts

L'apprentissage des messages courts est la deuxième étape du processus complet de la classification. Plusieurs types d'apprentissage existent dans la littérature. Nous nous intéressons dans ce chapitre à l'apprentissage supervisé, puisque nous sommes dans un cadre de classification supervisée.

Mathématiquement l'apprentissage supervisé peut être formulé comme suit : soit $D = \{(x_i, y_i)\}$ une base d'apprentissage indépendante et identiquement distribuée avec $x_i \in X$ et $y_i \in Y$.
X représente l'ensemble des données d'apprentissage de taille m (les messages dans notre cas). Y représente l'ensemble de classes possibles.
A chaque donnée de l'ensemble X est associée une classe de l'ensemble Y. L'ensemble D est utilisé pour trouver une fonction $f(x) = y$ qui prédit la classe y d'une nouvelle donnée x tout en optimisant un critère P. Ce critère sert à évaluer la fonction f.
Dans le domaine des messages courts, afin de construire le modèle de classification, les algorithmes traditionnels d'apprentissage supervisé ont été utilisés.
Plusieurs types d'algorithmes d'apprentissage supervisé existent dans la littérature. Nous proposons dans cette section l'état de l'art des algorithmes les plus populaires dans le domaine des messages courts.

2.4.1 Support Vector Machines

Les SVM ont été introduits par Cortes et Vapnik en 1995 [25]. Ils ont été conçus à la base pour la classification binaire où la classe d'un exemple x appartenant à la base d'apprentissage D fait partie de l'ensemble $Y = \{1, -1\}$. Ensuite, les SVM multi classes, où chaque classe est prédite par un SVM dédié, sont apparus.

Le principe général des SVM consiste à trouver l'hyperplan linéaire optimal qui permet de distinguer les données positives des données négatives de l'ensemble d'apprentissage. Cet hyperplan est représenté alors par la fonction hypothèse suivante : $h(x) = w^T x + w_0$ telle que $\forall\ 1 \leq i \leq m,\ y_i h(x_i) > 0$.
Soit w est le vecteur normal de $h(x)$, la distance entre un point x et l'hyperplan

séparateur est égale à $h(x)/||w||$ où $||w||$ est la norme euclidienne de w. Il existe plusieurs hyperplans possibles, il faut donc choisir le meilleur, celui qui coupe les deux nuages de données positives et négatives au juste milieu. Soit :

$$\arg\max_{w,w0} \; min||x - x_i|| : x \in \mathbb{R}^d, (wx + w_0) = 0, \; i = 1, ...m \tag{2.30}$$

C'est l'hyperplan qui maximise la distance minimale aux exemples d'apprentissage. La figure 2.14 résume le mécanisme des SVM. La recherche de l'hyperplan optimal

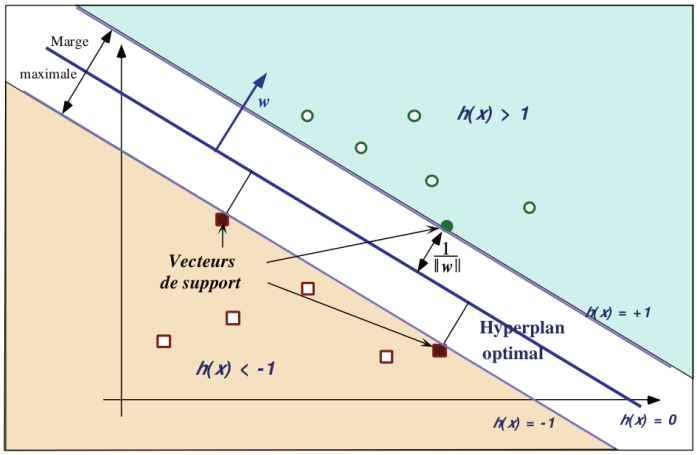


FIGURE 2.14 – les SVM : l'hyperplan optimal séparant linéairement les deux nuages de points [5].

revient à minimiser $||w||$, et donc à résoudre le problème d'optimisation suivant :

$$\begin{cases} Minimiser & \frac{1}{2}||w||^2 \\ sous\ les\ contraintes & u_i(w^T x + w_0) \geq 1 \; i = 1, ...m \end{cases} \tag{2.31}$$

Cette équation est appelée fonction primale. Nous pouvons l'écrire sous la forme d'une expression duale puisque la fonction objective de ce problème d'optimisation ainsi que les contraintes sont convexes. Pour résoudre ce type d'équations on utilise alors le *lagrangien* qui est défini comme étant une somme de la fonction objective et une combinaison linéaire des contraintes dont le coefficient $\alpha_i > 0$ est appelé *multiplicateur de lagrange* :

$$L(w, w_0, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i(u_i((x_i.w) + w_0) - 1) \tag{2.32}$$

Cette équation a une solution qui correspond au point-selle du lagrangien. A ce point, la dérivée du lagrangien s'annule. Le problème d'optimisation consiste alors à trouver les multiplicateurs de Lagrange tels que :

$$\begin{cases} Max_\alpha\{\sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i, \alpha_j u_i u_j(x_i, x_j)\} \\ \forall i, \alpha_i \geq 0 \\ \sum_i = 1^m \alpha_i u_i = 0 \end{cases} \tag{2.33}$$

Le problème d'optimisation que nous venons de décrire n'est applicable que dans le cas où l'on dispose d'un ensemble de données qui sont linéairement séparables. Afin de rendre la méthode plus souple, quelques erreurs sont tolérées au moment de la séparation des données positives des données négatives et du choix du meilleur hyperplan, et ceci par le biais des variables ressorts ε_i et une constante C qui représente un compromis entre la marge possible entre les données et le nombres d'erreurs acceptables.

Un autre type de travaux résolvant le problème des données non linéairement séparables consiste à passer à un autre espace de descripteurs de dimension plus grande où on peut trouver une séparation linéaire des données. Ce passage est effectué par des fonctions noyaux (exemple de noyaux : Linéaire, Polynomiale, Gaussien, Laplacien).

Les SVM ont été beaucoup utilisés dans plusieurs domaines et notamment dans le domaine de la classification des messages courts grâce à leurs bonnes performances. Nous les trouvons par exemple dans les travaux de Phan [59] et Chen [21] où ils ont été combinés avec des méthodes d'enrichissement basées sur les thèmes construits par LDA [13] et appliqués sur le corpus "search snippets" [59].
Les SVM ont été appliqués également dans les travaux de réduction des messages courts. Caragea par exemple, a utilisé les résultats obtenus par la classification des messages du corpus "Ushahidi" par les SVM pour comparer les différentes méthodes de représentation des données [18]. Yang dans [85] a montré que la performance des SVM combinés avec sa méthode d'abstraction est très élevée sur les deux corpus "search snippets" et "ohsumed". Wang et al. dans [82] ont également employé les SVMs comme algorithme d'apprentissage quand ils ont utilisé les concepts de Wikipédia pour améliorer la classification des textes courts.
Les SVM se sont montrés efficaces dans plusieurs travaux néanmoins, leur temps de calcul important pose un défi surtout dans les applications temps réel.

2.4.2 Entropie Maximum

Entropie Maximum (MaxEnt) a été proposé par Berger en 1996 [7]. C'est une méthode qui est largement utilisée dans le domaine de Traitement Automatique de la Langue telle que la reconnaissance des entités nommées et l'identification des parties de discourt des textes. Voici son principe de base :
Comme tout algorithme d'apprentissage supervisé, nous disposons d'une base d'apprentissage $\{(x_1, y_1), (x_2, y_2), .., (x_N, y_N)\}$, $x \in X$ étant l'ensemble de phrases et $y \in Y$ étant l'ensemble de sorties. Cette base peut être exprimée en terme de distribution empirique de probabilité comme suit :

$$\tilde{p}(x, y) = \frac{1}{N} \times nombre\ d'occurneces\ de\ (x, y) \tag{2.34}$$

Le but est de construire un modèle statistique qui génère l'exemple d'apprentissage $\tilde{p}(x, y)$ en appliquant sur ce dernier un ensemble de statistiques. Si nous cherchons

par exemple un modèle pour traduire un mot de l'anglais en français, la fréquence de traduction du mot "in" en "en" est $\frac{3}{10}$. Un autre exemple qui tient en compte le contexte de x est : si le mot "in" est suivi par le mot "April", la probabilité de le traduire en "en" est de 90%. Pour exprimer cette statistique, soit f(x,y) une fonction binaire de (x,y) qu'on appelle ***caractéristique***. Elle est définie comme suit :

$$f(x,y) \begin{cases} 1 \ si \ y = "en" \ suivie \ par \ "April" \\ 0 \ sinon \end{cases} \tag{2.35}$$

La valeur attendue de f par rapport à la distribution empirique est donc :

$$\tilde{p}(f) = \sum_{x,y} \tilde{p}(x,y) f(x,y) \tag{2.36}$$

La valeur attendue de f par rapport au modèle est :

$$p(f) = \sum_{x,y} \tilde{p}(x) p(y|x) f(x,y) \tag{2.37}$$

La valeur attendue de f par le modèle doit être celle obtenue empiriquement avec la base d'apprentissage. D'où :

$$p(f) = \tilde{p}(f) \tag{2.38}$$

On appelle cette égalité une ***contrainte*** qui doit être respectée au moment où l'on choisit le modèle. Pour généraliser nous disposons maintenant de n fonctions caractéristiques f_i et nous voulons chercher le modèle optimal p^* respectant toutes les contraintes générées par ces n fonctions. Soit P l'ensemble de tous les modèles possibles. Soit p un modèle appartenant à C un sous ensemble de P.

$$C = \{p \in P | p(f_i) = \tilde{p}(f_i) \ \ \forall i \in \{1, 2, ..., n\}\} \tag{2.39}$$

Comme l'intersection de toutes les contraintes donne un ensemble infinie de modèles, le principe de cette méthode consiste à choisir le modèle dont la distribution est la plus uniforme. Cette uniformité est donnée par l'entropie conditionnelle suivante :

$$H(p) = -\sum_{x,y} \tilde{p}(x) \ p(y|x) \ log(p(y|x)) \tag{2.40}$$

Le modèle dont la distribution est la plus uniforme est le modèle qui a l'entropie maximale $H(p)$, autrement dit, cette méthode consiste à choisir le modèle $p^* \in C$ qui maximise $H(p)$. Soit :

$$p^* = \arg\max_{p \in C} H(p) \tag{2.41}$$

Il existe toujours un seul modèle p^* respectant toutes les contraintes dans C qui a une entropie maximale.

Vu son utilisation avec succès dans le domaine du TAL, maxEnt a aussi été utilisé pour la classification des messages courts notamment dans les travaux d'enrichissement de ces derniers [59], [21]. Les résultats des expérimentations sur ces travaux ont montré que maxEnt combiné avec les différentes méthodes d'enrichissement est efficace. Cette efficacité est due principalement aux méthodes d'enrichissement mais aussi à la robustesse de MaxEnt ainsi qu'à sa rapidité.

2.4.3 Naive Bayes

Naive Bayes [69] est un algorithme d'apprentissage supervisé, il appartient à la catégorie des classifieurs probabilistes. Il est basé sur le théorème de Bayes et sur l'hypothèse de l ?indépendance entre les caractéristiques. Étant donnée une variable de classe y et un ensemble de caractéristiques $x_1, x_2, .., x_n$, Naive Bayes considère que la probabilité d'une classe est y pour un exemple donné est :

$$P(y|x_1, ..x_n) = \frac{P(y)P(x_1, x_n|y)}{P(x_1, .., x_n)} \tag{2.42}$$

En se basant sur l'hypothèse que $x_1, x_2, .., x_n$ sont conditionnellement indépendants, La probabilité peut être simplifiée à :

$$P(y|x_1, ..x_n) = \frac{P(y) \times \prod_{i=1}^{n} P(x_i|y)}{P(x_1, .., x_n)} \tag{2.43}$$

La classe finale $\widehat{y}$ est donnée par :

$$\widehat{y} = \arg\max P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.44}$$

$P(y)$ peut être estimée en tenant compte de sa fréquence dans la base d'apprentissage.

Trois implémentations populaires de Naive Bayes existent dans la littérature. Elles diffèrent dans la façon d'estimer $P(x_i|y)$. En effet, sa distribution peut être Gaussienne, multinomiale ou de Bernoulli.

Bien que cet algorithme d'apprentissage se base sur une hypothèse naïve, il a été démontré efficace dans le domaine de la classification de textes, essentiellement pour la détection des spam dans les messages électroniques [4] [52]. C'est un algorithme rapide et ne nécessite pas beaucoup de données d'apprentissage. Naive Bayes a été également utilisé dans la classification des textes courts. Sriram et al. dans [74] par exemple ont l'appliqué sur leur nouvelle représentation matricielle des tweets et ils ont obtenu des résultats plus performants que ceux obtenus par le BOW. Wang et al dans [80] ont utilisé le Naive Bayes multinomial avec leur nouvelle approche. Cette dernière consiste à construire un ensemble de thésaurus de caractéristiques dont les poids sont forts en se basant sur LDA [13] et sur le gain d'information. Les résultats obtenus montrent que cette approche est plus efficace que les algorithmes traditionnels tels que les SVM.

2.5 Classification des messages courts

Afin de classer les messages courts, on utilise généralement le modèle de classification qui a été construit pendant la phase d'apprentissage. Néanmoins, il y a des travaux qui s'inspirent du domaine de la "recherche d'information" pour classer les messages dans des groupes prédéfinis. Sun, par exemple, dans [76], propose une plate-forme résumée dans la figure 2.15.

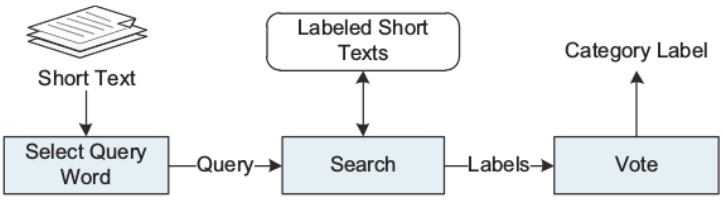


FIGURE 2.15 – Plate-forme proposée par Sun dans [76] pour la classification des messages courts.

Pour classer un nouveau message, la première étape consiste à le transformer en une requête de mots pondérés puis d'envoyer cette requête à un moteur de recherche local. Ce dernier recherche les exemples les plus proches de la requête dans la base d'apprentissage. La classe finale du message est déterminée par un vote majoritaire sur les classes de l'ensemble des messages retournés.

Caragea, dans [18] a utilisé une méthode de classification par mots clés pour classer les messages du corpus "Ushahidi". La classification par mots clés consiste à identifier pour chaque catégorie les k mots dont la fréquence d'occurrence est la plus élevée. Ces k mots sont considérés comme les mots clés de cette catégorie. Un message m appartient à une catégorie c s'il contient au moins un mot clé de cette catégorie (on considère qu'il n'y a pas de chevauchement entre les mots clés des catégories). Bien que cette méthode soit simple dans sa construction, les expérimentations faites par Caragea ont montré qu'elle produit des résultats comparables avec ceux obtenus par les SVM.

Wang et al. dans [81] propose une nouvelle approche pour la classification et le classement des textes courts qu'ils appellent "BocSTC". Le principe générale de cette approche consiste à transformer la représentation traditionnelle des messages courts "Bag of Words" en une base de concepts "Bag of Concepts". Puis, d'appliquer des mesures de similarité entre le message à classer et l'ensemble de concepts générés pour prédire la classe du message.
Pour la création de la base de concepts, Wang et al. utilisent la base de connaissance Probase [84]. La première étape consiste à extraire les différentes entités de chaque texte en utilisant la segmentation et le stemming, puis, pour chaque classe CL_l, un ensemble de concepts candidats est généré en utilisant les entités extraites et essentiellement les relations de type " est - un" dans Probase. Un poids w_i est associé à chaque concept candidat c_i par classe en utilisant les deux mesures idf et Typicité. Cette dernière permet de mesurer la typicité d'un concept c par rapport à l'ensemble des contextes contenant une entité e $(P(e|c))$. Les concepts qui ont un poids faible sont supprimés. Une classe à la fin est représentée sous forme d'un vecteur de concepts $CL_l = (< C_1, w_1 >, .., < C_i, w_i >, .., < C_k, w_k >),\ i = 1, 2, .., k$. Le nouveau message à classer nommé $st_i = \{e_j, j = 1, 2, .., M\}$ est lui même transformé en un vecteur de concepts $C_{st_i} = \{C_j, j = 1, 2, .., M\}$. La similarité entre un

message st_i et une classe CL_l est donnée par la formule 2.45

$$Sim(st_i, CL_l) = \sum_{C_j} \sum_{c_k \in C_j \cap c_k \in CL_l} P(c_k|e_j) \times w(c_k, CL_l) \tag{2.45}$$

Le vote final est attribué à la classe qui possède le score le plus élevé. L'approche a été testée pour la classification ainsi que le classement des messages courts dans le cadre d'une application réelle consistant à proposer des canaux télévisés basés sur des requêtes de recommandations. BocSTC surpasse les méthodes traditionnelles telles que les SVM en terme précision et de rappel.

Zelikoviz et Hirsh dans [89] proposent une nouvelle approche qui associe une base de documents externes à la base d'apprentissage pour classer les messages. L'approche se base sur le moteur de recherche WHIRL [24] utilisé surtout pour la recherche de l'information textuelle. Afin de trouver les messages labellisés les plus proches d'un message à classer comme dans l'approche de Sun [76], une requête WHIRL est formulée. Un exemple de requête est le suivant :

SELECT Test.instance,Train.label
From Train AND Test
WHERE Train.instance SIM Test.instance

Comme résultat de cette requête, le moteur de recherche retourne une liste de K tuples sous forme :

$$< Test.instance, Train.instance, Train.label >$$

K étant un paramètre défini par l'utilisateur. La liste des tuples retournée par WHIRL est ordonnée selon un score. Le premier élément dans la liste représente le message le plus proche du message requête. La similarité entre les documents est calculée par la mesure "cosine".
La nouvelle méthode proposée par Zelikoviz et Hirsh vise à utiliser l'ensemble des documents externes comme un pont sémantique entre le message à classer et la base d'apprentissage. La nouvelle requête sera donc :

SELECT Test.instance,Train.label
From Train AND Test AND Background
WHERE Train.instance SIM Background.value AND Test.instance SIM Background.value

Deux scores issus des deux similarités (base d'apprentissage, documents externes et documents externes, exemple de test) sont générés pour chaque tuple. WHIRL multiplie les deux scores pour calculer le score final. Le label attribué à l'exemple d'apprentissage qui a le score le plus élevé est considéré comme la classe recherchée. Ainsi, cette requête nous permet de relier sémantiquement les messages courts à classer avec des exemples de la base appartenant à la même catégorie même s'ils ne partagent pas des mots communs.

2.6 Validation de la classification des messages courts

Une fois le modèle de classification créé, il est obligatoire de le tester et de le valider avant de le mettre en application réelle.
Pour tester un modèle, on calcule la valeur d'une mesure sur un ensemble de données dont on connaît les classes réelles. Le calcul se base sur la comparaison de la valeur de la classe réelle et la valeur de la classe prédite par le modèle. Cette mesure est appelée une métrique d'évaluation. Dans cette section nous résumons les métriques qui ont été utilisées pour tester les modèles de classification des messages courts.

2.6.1 Accuracy

D'une façon générale, l'accuracy est le ratio entre les exemples bien classés et le nombre total d'exemples.
Dans le cas de la classification binaire (positif / négatif), l'accuracy est calculée à partir de la matrice de confusion suivante :

TABLE 2.2 – Matrice de confusion des classes prédites par le modèle et les classes réelles des données dans le cas binaire

	positif prédit	**négatif prédit**
positif	*# vrai positif*	*# faux négatif*
négatif	*# faux positif*	*# vrai négatif*

L'accuracy est définie par le ratio suivant :

$$\frac{\#\ vrai\ positif\ + \#\ vrai\ négatif}{\#\ vrai\ positif\ + \#\ faux\ positif\ +\ \#\ vrai\ négatif +\ \#\ faux\ négatif} \tag{2.46}$$

Dans le cas des multi-classes, la table de confusion est généralisée sur plusieurs classes. Soit A une matrice de confusion, l'élément $A_{(i,j)}$ représente le nombre de fois où le modèle à prédit la classe j alors que la vraie classe est i. L'accuracy est donc :

$$Accuracy = \frac{\sum_i A_{ii}}{\sum_i \sum_j A_{i,j}} \tag{2.47}$$

Dans la classification des messages courts, l'accuracy a été utilisée par Phan par exemple dans [59], Chen dans [21], Yang dans [85] et Sriram dans [74] pour valider les différents travaux proposés.

2.6.2 Erreur de classification

L'erreur de classification est l'inverse de l'accuracy. Elle est définie comme étant le ratio entre le nombre d'exemples mal classés et le nombre total d'exemples. Si nous reprenons la matrice de confusion de la table 2.2, l'erreur de classification pour

le cas binaire sera donc :

$$\frac{\#\ faux\ positif\ + \#\ faux\ négatif}{\#\ vrai\ positif\ + \#\ faux\ positif\ +\ \#\ vrai\ négatif +\ \#\ faux\ négatif} \tag{2.48}$$

et pour le cas multi-classes :

$$Erreur\ de\ classification = \frac{\sum_{i \neq j} A_{ij}}{\sum_i \sum_j A_{i,j}} \tag{2.49}$$

L'erreur de classification a été également utilisée par Phan et Chen dans respectivement [59] et [21].

2.6.3 F1-mesure

Le calcul de cette mesure se base sur deux autres mesures qui sont la précision et le rappel. La F1-mesure est une moyenne harmonique entre ces deux dernières. En tenant compte de la Matrice de confusion 2.2, la précision est :

$$P = \frac{\#\ vrai\ positif}{\#\ vrai\ positif\ +\ \#\ faux\ positif} \tag{2.50}$$

Le rappel est :

$$R = \frac{\#\ vrai\ positif}{\#\ vrai\ positif\ +\ \#\ faux\ négatif} \tag{2.51}$$

F1-mesure est donc :

$$F1 - mesure = \frac{2PR}{P + R} \tag{2.52}$$

La F-1 mesure a été utilisée par Caragea dans [18] pour comparer les différentes méthodes de représentation des messages courts.

2.6.4 Validation croisée

La validation croisée est une technique d'évaluation de l'efficacité d'un modèle en se basant sur l'échantillonnage. Cette technique est très utile surtout dans le cas où l'on ne dispose pas d'un ensemble de données de test. Trois grands types de validation croisée existent dans la littérature :

— **holdout method** : c'est la méthode la plus simple. Elle consiste à décomposer aléatoirement l'ensemble de données en deux parties, apprentissage et test, avec un pourcentage plus grand pour la partie apprentissage.

— **k-fold cross-validation** : il s'agit de répartir la base de données en k sous ensembles, puis d'en choisir un parmi ces k sous ensembles pour le test. Les $k-1$ restant sont utilisés pour l'apprentissage. Cette opération est répétée k fois afin que chaque sous ensemble soit utilisé une fois pour le test.

— **leave-one-out cross-validation** : cette méthode est un cas particulier de la méthode k-fold cross-validation. Si nous supposons que le nombre total de données est égal à n, cette méthode consiste à apprendre sur les $n-1$ éléments et tester le modèle créé sur le $n_{ème}$. Cette opération se répète n fois.

Une fois que la répartition des données est faite, une mesure de performance telle que l'accuracy est appliquée sur les ensembles obtenus. Une moyenne de tous les scores obtenus est générée pour le cas de k-fold et leave-one-out. Cette moyenne représente le score final. La validation croisée a été beaucoup utilisée dans les travaux de classification des messages courts. Parmi les trois méthodes existantes, la méthode k-fold-CV est la plus utilisée, Phan a utilisé 5-fold-CV avec la mesure erreur de classification [59]. Caragea a utilisé aussi 5-fold-CV combiné cette fois ci avec F1-mesure [18].
Un autre type d'échantillonnage a été proposé dans l'état de l'art des messages courts. Il s'agit de construire des sous-ensembles de données de tailles différentes à partir de la base initiale puis de construire des différents modèles de classification sur ces différents sous-ensembles. Le but est de montrer que les méthodes proposées sont efficaces même sur des bases de données de petite taille [59] [85].

2.7 Synthèse

Ce chapitre présente un résumé de l'état de l'art du processus complet de la classification des messages courts. Par rapport à la classification des textes traditionnels, ce type de classification présente des défis additionnels. En effet, la courte taille de ce type de texte, la parcimonie et le bruit introduit dans ces messages font que les modèles de classification créés par les méthodes traditionnelles ne sont pas très efficaces et les mesures de similarité échouent à trouver les vrais liens existant entre les messages. Pour résoudre ces différentes lacunes, les chercheurs se sont concentrés sur l'amélioration de la qualité des messages. Nous avons distingué deux grands types de travaux. Une première communauté s'est dirigée vers l'enrichissement, par la connaissance externe, des messages courts. Plusieurs façons d'enrichir les messages ont été identifiées dans la littérature. Il y a des travaux qui utilisent les ontologies comme source d'enrichissement telles que WordNet, Mesh et Wikipédia. Hu et Zang par exemple ont utilisé dans [40] Wikipédia pour enrichir les textes initiaux par ses concepts (les titres des articles) et ses catégories. Hu et Sun dans [39] ont proposé un algorithme de génération de caractéristiques à partir de Wikipédia et WordNet pour enrichir le texte d'origine.
Un autre type d'enrichissement a été identifié. Il se base sur les techniques du "topic model" telles que LSI (LSA) [26], PLSI (PLSA) [35] ou LDA [13]. Il s'agit de collecter un ensemble de documents externes qui sont sémantiquement proche de la base d'apprentissage initiale puis d'appliquer sur ces documents une des techniques du topic model afin d'obtenir plusieurs regroupements de mots. Chaque regroupement représente une thématique particulière. Il s'agit ensuite d'enrichir les différents messages par les thématiques les plus proches comme mentionné dans [59] et [21].

Un deuxième axe de recherche pour améliorer la qualité des messages se concentre sur la réduction de la matrice représentative des messages pour réduire la parcimonie des données. Une des méthodes de réduction des messages est l'abstraction, elle considère les termes similaires comme une seule entité et les remplace par le terme

le plus abstrait. L'information dans ce cas est condensée dans une matrice plus réduite. Caragea par exemple dans [18] a utilisé l'algorithme d'abstraction proposé par Silvescu dans [72] pour classer des messages courts. Les techniques du topics model ont été aussi présentes pour réduire la taille des messages, Yang par exemple a réduit l'ensemble des mots composant son corpus aux thèmes construits par LDA. [85]. Zelikoviz a réduit sa matrice de données en utilisant LSI [88]
La sélection de caractéristiques était aussi mise en œuvre dans la classification des messages courts. Caragéa [18] par exemple a utilisé l'algorithme de relief [43] pour choisir les caractéristiques pertinentes. Sun [76] a sélectionné ses caractéristiques par le biais de la clarté comme mesure de similarité de documents. Sriram à son tour a utilisé la sélection des caractéristiques à partir des méta données des tweets [74].

Une fois que la qualité des messages est amélioré soit par enrichissement soit par réduction, l'étape suivante consiste à appliquer un algorithme d'apprentissage supervisé pour créer le modèle de classification. Deux algorithmes se détachent pour la classification des messages courts, les SVM [25] dont le principe est de trouver l'hyperplan optimal qui permet de mieux séparer les données positives de celles négatives, et Maximum Entropy qui consiste à trouver le modèle optimal dont l'entropie est maximale et qui a été utilisé avec succès dans le domaine de Traitement Automatique de la Langue [7].

Pour classer les messages les chercheurs se sont inspirés aussi du domaine de la recherche d'information comme dans [76] et dans [88]. Une requête est formulée à partir du message à classer puis est envoyée dans un moteur de recherche. Le résultat de la recherche est l'ensemble des documents qui sont proches du message à classer. La classe finale est donnée, par exemple, par un vote majoritaire.

Une fois le modèle créé, il doit être validé pour l'utilisation réelle sur les messages à classer. Plusieurs métriques d'évaluation ont été utilisées pour la validation du modèle telles que par exemple l'accuracy, l'erreur de classification, la F1-mesure et la validation croisée.

Pour conclure, l'étude du processus complet de la classification des messages courts montre que les travaux existants se sont concentrés sur la phase de prétraitement. Les recherches utilisent souvent les algorithmes classiques pour la phase d'apprentissage. A notre connaissance aucune recherche n'a proposée un algorithme qui soit spécifique aux messages courts. Le travail réalisé dans cette thèse va intervenir à la fois dans la phase de prétraitement et la phase d'apprentissage. Le but est d'améliorer la qualité des messages suivant la même ligne que les anciens travaux et de proposer un nouvel algorithme d'apprentissage plus adapté à ce type de texte.

D'un autre côté, le problème de classification des messages courts est généralement traité d'une vision statique où une fois le modèle de classification créé, il ne change plus au court du temps. Un processus dynamique sera proposé pour mettre à jour le modèle de classification de base en profitant de l'expertise de l'utilisateur.

Chapitre 3

Forêts aléatoires

Sommaire

3.1 Introduction 45
3.2 Arbres de décision 46
3.3 Bagging 50
3.4 Random Feature Selection 51
3.5 Force et corrélation 52
3.6 Consistance des forêts aléatoires 53
3.7 Types de forêts aléatoires 55
3.8 Forêts aléatoires et apprentissage dynamique 56
3.9 Synthèse 61

3.1 Introduction

Les forêts aléatoires, ou les "Random Forest" (RF), sont un algorithme d'apprentissage supervisé introduit par Breiman dans [15]. Elles ont connu beaucoup de succès dans le domaine de l'apprentissage automatique depuis 2001 jusqu'à nos jours. Cet énorme succès est dû principalement à leurs très bonnes performances dans plusieurs domaines. Breiman les définit comme suit :
Il s'agit d'un ensemble de classifieurs de type arbre de décision [16] notés :

$$\{h(x, \theta_k) \quad k = 1, ...L\}$$

Où θ_k est une famille de vecteurs aléatoires indépendants et identiquement distribués. Chaque arbre d'une forêt aléatoire donne une classe pour une donnée d'entrée x, la classe retenue au final est déduite par vote majoritaire de tous les arbres.
Les forêts aléatoires sont conçues principalement pour améliorer la performance de prédiction des données en partant du principe que considérer l'avis de plusieurs experts est meilleur que l'avis d'un seul. Elles sont également développées pour remédier aux limites des arbres de décision. L'avantage des forêts aléatoires se situe dans l'augmentation du caractère aléatoire dans leur construction. Cette dernière, en effet se base sur deux méthodes d'induction d'un ensemble de classifieurs qui sont le Bagging [14] et le Random Feature Selection.

L'état de l'art des forêts aléatoires est très riche, ce chapitre va rappeler les grands principes de cette méthode en les illustrant à chaque fois par un exemple de

notre domaine de recherche : les messages courts. Nous présentons ainsi les arbres de décisions, le Bagging et le processus de sélection de caractéristiques. Nous présentons également les propriétés de convergence et de consistance des forêts aléatoires. Plusieurs travaux ont dérivé à partir de l'implémentation de base des RF vers une multitude de variantes. Les plus populaires sont citées dans ce chapitre. Les forêts aléatoires ont aussi été adaptées pour répondre à des exigences particulières dans le domaine de l'apprentissage en ligne, la dernière partie de ce chapitre sera consacrée à ce sujet.

3.2 Arbres de décision

Les arbres de décision font partie des algorithmes supervisés dans le domaine de l'apprentissage automatique. Leur principe consiste à partitionner les données d'apprentissage en des groupes dont le contenu est de plus en plus homogène jusqu'à l'obtention de données pures (appartenant à la même classe) ou l'atteinte d'un nombre maximum de partitionnement. Le modèle résultat est un arbre composé de plusieurs règles de décisions et qui est facilement interprétable.

Afin de mieux comprendre le principe des arbres de décision dans le cadre des messages courts nous reprenons l'exemple proposé par Deerwester dans [26] pour montrer l'efficacité de sa méthode LSI. L'exemple contient 9 titres de documents classés en deux catégories (IHM et graphes). Nous considérons ces titres comme un exemple simplifié d'une base d'apprentissage. La matrice représentative de ces 9 titres est résumée dans la figure 3.1. Pour faire plus simple, l'espace de caractéristiques est réduit de la totalité des mots composant tous ces titres à 10 mots seulement.

titre	human	interface	computer	user	system	response	time	EPS	survey	minors	classe
c1	1	1	1	0	0	0	0	0	0	0	IHM
c2	0	0	1	1	1	1	1	0	1	0	IHM
c3	0	1	0	1	1	0	0	1	0	0	IHM
c4	1	0	0	0	2	0	0	1	0	0	IHM
c5	0	0	0	1	0	1	1	0	0	0	IHM
m1	0	0	0	0	0	0	0	0	0	0	graphes
m2	0	0	0	0	0	0	0	0	0	0	graphes
m3	0	0	0	0	0	0	0	0	0	1	graphes
m4	0	0	0	0	0	0	0	0	1	1	graphes

FIGURE 3.1 – Matrice représentative des 9 titres de documents [26].

Pour construire chaque nœud de l'arbre, l'algorithme cherche à chaque fois la meilleure caractéristique permettant de répartir les données le plus purement possible en fonction de ses valeurs et les classes des exemples. Les valeurs des différentes caractéristiques dans cette exemple sont leur nombre d'occurrences dans chaque texte. Le nœud peut séparer les données en deux branches : Une première branche pour les textes où le nombre d'occurrences est égal à 0, une deuxième branche où ce nombre est supérieur à 0 pour dire que le mot se produit une ou plusieurs fois dans le texte. Si nous prenons par exemple la caractéristique "user", nous remarquons ici qu'aucun texte de la catégorie graphes ne contiennent pas ce mot, mais aussi

deux parmi 5 titres de la catégorie "IHM" ne le contiennent pas non plus. Pour la branche où le nombre d'occurrences de cette caractéristique est null, il est impossible à ce stade de conclure la classe à prédire. Il faut donc continuer à partitionner et construire de nouveaux nœuds dans l'arbre en utilisant d'autres caractéristiques. Par contre, pour le cas de la deuxième branche où le nombre d'occurrences est strictement positif, nous sommes certains de dire que la classe recherché est "IHM". Le nœud suivant est donc le nœud terminal qui est aussi appelé feuille de l'arbre. La classe "IHM" est attribuée à cette feuille. La figure 3.2 montre un arbre généré à partir de la base des titres en utilisant J48, une implémentation libre de l'algorithme C4.5 en java [1].

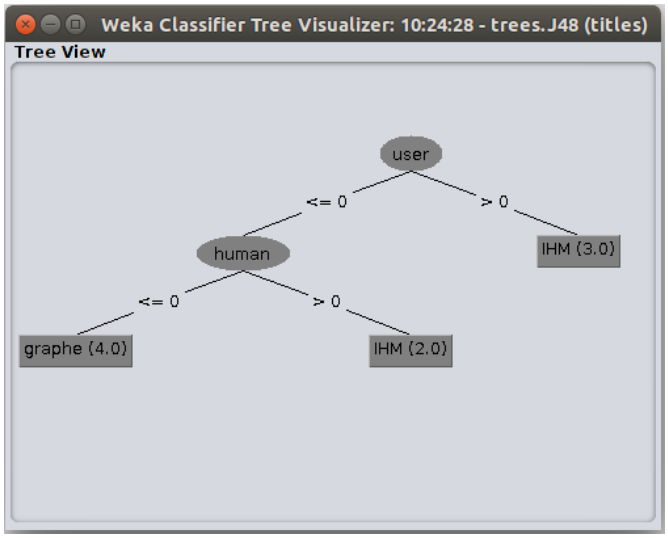


FIGURE 3.2 – Modèle d'arbre généré à partir de la base de titres.

Pour classer un nouveau titre, un chemin va être parcouru parmi les chemins de l'arbre jusqu'à l'arrivée à une feuille : la décision peut être prise à ce moment. On appelle la longueur du chemin le plus long dans l'arbre sa profondeur. Si aucune indication n'existe sur une taille maximale, toutes les feuilles construites à la fin sont issues d'un nœud pur.
Pour choisir un chemin, nous nous plaçons à la racine de l'arbre. Si le titre contient au moins une occurrence de "user" la classe est alors "IHM", sinon s'il contient au moins une occurrence de "human" la classe est "IHM" aussi, sinon la classe est "graphes".

Nous remarquons ici que parmi les 10 caractéristiques disponibles nous arrivons à décider en ne tenant compte que de deux seulement, mais comment choisir les caractéristiques pour que nous puissions à la fois minimiser le nombre de nœuds et décider le plus correctement possible ?
Pour choisir la meilleure caractéristique, i.e, la variable la plus discriminante, nous

1. La bibliothèque Weka (http ://www.cs.waikato.ac.nz/ml/weka/) est utilisée pour générer l'arbre.

calculons le gain d'information de chacune en utilisant un critère de partitionnement. Deux critères sont généralement utilisés, l'indice de Gini utilisé dans CART [16] et l'entropie utilisée par exemple dans C4.5 [61], deux implémentations des arbres de décision très connues dans la littérature.
Soit S un échantillon représentant toutes les valeurs que peut prendre une caractéristique (une colonne dans la matrice des messages). Soit $S_i,\ i = 1..k$ une partition de S selon les classes prédéfinies de la base d'apprentissage. Le critère de Gini est calculé comme suit :

$$Gini(S) = \sum_{i=1}^{k} \frac{|S_i|}{|S|} \times (1 - \frac{|S_i|}{|S|}) \tag{3.1}$$

L'entropie est calculée comme suit :

$$Entropie(S) = -\sum_{i=1}^{k} \frac{|S_i|}{|S|} \times \log_2(\frac{|S_i|}{|S|}) \tag{3.2}$$

Soit T une caractéristique candidate ou un test potentiel. T peut prendre deux ou plusieurs valeurs (n), chaque valeur représentant une branche. Le gain d'information en utilisant T comme test potentiel et en utilisant Gini ou l'entropie comme critère d'évaluation est calculé respectivement selon les formules 3.3 et 3.4.

$$Gain_{gini}(S_p, T) = Gini(S_p) - \sum_{j=1}^{n} P_j \times Gini(S_{p_j}) \tag{3.3}$$

$$Gain_{entropie}(S_p) = Entropie(S_p) - \sum_{j=1}^{n} P_j \times Entropie(S_{p_j}) \tag{3.4}$$

S_p est l'échantillon associé à la position p dans l'arbre et P_j représente la portion satisfaisant la valeur j du test T. Nous remarquons ici que le premier terme de chacune de ces deux formules ne dépend pas de la caractéristique T. Maximiser le gain d'information revient donc à minimiser le deuxième terme de chacune des deux. Un indice de Gini ou une entropie élevée veulent dire que la répartition en utilisant la caractéristique T aboutit à un groupement hétérogène dont les classes sont mélangées, le niveau de pureté est très bas dans ce cas. Par contre, dans le cas où l'indice de Gini ou l'entropie sont bas, T est alors discriminative.

Calculons par exemple le gain d'information de la caractéristique "user" de la base des titres. Supposons que p = 1, que nous sommes à la racine de l'arbre et que nous travaillons sur la totalité des données. Deux branches existent : soit le nombre d'occurrences d'une caractéristique $<= 0$, soit > 0, alors $n = 2$.

Indice de Gini

- $Gini(S) = \frac{5}{9} \times \frac{4}{9} + \frac{4}{9} \times \frac{5}{9} = 0.493$
- $Gain_{gini}(S_1, "user") = Gini(S) - (\frac{6}{9} \times Gini(s_{(<=0)}) + \frac{3}{9} \times Gini(s_{(>0)}))$
- $Gain_{gini}(S_1, "user") = 0.493 - (\frac{6}{9} \times \frac{2}{6} \times \frac{4}{6} \times 2 + \frac{3}{9} \times \frac{3}{3} \times \frac{0}{3})$
- $Gain_{gini}(S_1, "user") = 0.197$

Entropie

— $Entropie(S) = -\frac{5}{9}\log_2(\frac{5}{9}) - \frac{4}{9}\log_2(\frac{4}{9}) = 0.991$
— $Gain_{entropie}(S_1, "user") = Entropie(S) - (\frac{6}{9} \times Entropie(s_{(<=0)}) + \frac{3}{9} \times Entropie(s_{(>0)}))$
— $Gain_{entropie}(S_1, "user") = 0.298 - (\frac{6}{9} \times (-\frac{2}{6}\log_2(\frac{2}{6}) - \frac{4}{6}\log_2(\frac{4}{6})) + \frac{3}{9} \times (-\frac{3}{3}\log_2(\frac{3}{3}))$
— $Gain_{entropie}(S_1, "user") = 0.378$

Que ce soit en optant pour le critère de Gini ou pour l'entropie, les gains d'information des 9 autres caractéristiques sont calculés de la même manière. le tableau 3.1 résume les gains obtenus pour toutes les caractéristiques.

TABLE 3.1 – Gains d'information de toutes les caractéristiques de la base des titres

caractéristique	**GINI**	**Entropie**
"human"	0.113	0.224
"interface"	0.113	0.224
"computer"	0.113	0.224
"user"	0.197	0.378
"system"	0.197	0.378
"response"	0.113	0.224
"time"	0.113	0.224
"EPS"	0.113	0.224
"survey"	0.001	0.002
"minors"	0.176	0.319

les gains obtenus sont ensuite comparés pour trouver la caractéristique la plus discriminante qui est dans ce cas "user" car elle a le gain d'information le plus grand. Cette caractéristique est utilisée alors pour construire le nœud racine comme le montre la figure 3.2.

Les arbres de décision présentent plusieurs avantages : il s'agit en effet d'un modèle non paramétrique, facilement compréhensible et traduisible en une base de règles. Il supporte bien le multi-classes et il est insensible aux variables redondantes. Mais ses meilleurs performances ne sont atteintes que lorsqu'il est appliqué sur des bases de données volumineuses, dans le cas contraire, le modèle peut souffrir d'instabilité. Les arbres de décisions sont connus également par leur forte dépendance vis-à-vis des données d'apprentissage et par la forte probabilité de tomber dans le sur-apprentissage. Pour contourner ces limites, la notion de forêts aléatoires [15] est introduite. A partir de la même base d'apprentissage, au lieu d'induire un seul arbre de décision, le concept des forêts aléatoires consiste en induire plusieurs. Une multitude de méthodes existent dans la littérature pour construire un ensemble de classifieurs. Comme nous l'avons déjà mentionné, Breiman en a utilisé deux pour la constructions des arbres de décisions de ses forêts : le bagging et le "Random Feature Selection". Nous détaillons dans les deux paragraphes suivants ces deux principes.

3.3 Bagging

Le bagging a été proposé par Breiman dans [14] en 1996. Son idée de base est de générer, à partir d'une base d'apprentissage B, un ensemble de classifieurs différents et de les agréger ensuite dans un seul à l'aide un vote majoritaire par exemple. Le bagging se repose sur le principe du bootstrap, d'où son appellation (**B**ootstap **agg**rega**ting**). Il s'agit d'une technique d'inférence statistique qui date des années 70 et qui consiste à ré-échantillonner un ensemble d'observations. Les différents sous échantillons obtenus peuvent être ensuite analysés ou utilisés pour faire des tests statistiques. Dans le cas des forêts aléatoires, Breiman a utilisé la notion du bootstrap pour construire des sous ensembles d'exemples à partir de la base d'apprentissage B. Chaque sous ensemble B_i contient des exemples qui sont tirés aléatoirement et avec remise et donne naissance à un classifieur qui est un arbre de décision.

Nous avons présenté le Bagging comme une méthode de combinaison de plusieurs classifieurs, il est cependant aussi efficacement utilisé dans le domaine de la régression. Les expérimentations menées par Breiman sur différents benchmarks réels ou simulés ont montré une augmentation de l'accuracy. Breiman a aussi prouvé que le bagging fonctionne bien même en utilisant des classifieurs instables comme les arbres de décision.

Pour illustrer le principe du bagging dans le cadre de la classification des messages courts, nous reprenons l'exemple de la base des titres de la figure 3.1. Le but est ici de construire plusieurs arbres au lieu d'un seul. Un exemple de tirage aléatoire et avec remise est présenté dans la figure 3.3.

titre	human	interface	computer	user	system	response	time	EPS	survey	minors	classe
c1	1	1	1	0	0	0	0	0	0	0	IHM
c2	0	0	1	1	1	1	1	0	1	0	IHM
c3	0	1	0	1	1	0	0	1	0	0	IHM
c4	1	0	0	0	2	0	0	1	0	0	IHM
c5	0	0	0	1	0	1	1	0	0	0	IHM
m1	0	0	0	0	0	0	0	0	0	0	graphes
m2	0	0	0	0	0	0	0	0	0	0	graphes
m3	0	0	0	0	0	0	0	0	0	1	graphes
m4	0	0	0	0	0	0	0	0	1	1	graphes

FIGURE 3.3 – Illustration du bagging sur la base de titres.

Nous remarquons, à partir de la figure 3.3, que nous avons choisi 6 exemples parmi les 9 disponibles, les vecteurs en rose représentent les lignes sélectionnées. Les lignes c_2 et m_4 sont en gras pour dire qu'elles ont été choisies deux fois. La nouvelle base d'apprentissage ainsi que l'arbre généré à partir de cette base sont présentés dans la figure 3.4.

Nous remarquons, à partir de l'exemple, que la caractéristique "user" est devenue fortement discriminante puisque la partition issue de cette variable contient deux nœuds purs, comme on peut le voir dans l'arbre généré. En effet, les exemples contradictoires n'ont pas été choisis dans ce bootstrap. Le risque de se tromper en classant les nouveaux titres avec cet arbre tout seul est plus grand qu'en les classant avec l'arbre 3.2, car un titre n'ayant pas le mot "user", peut appartenir aussi à la

titre	human	interface	computer	user	system	response	time	EPS	survey	minors	classe
c2	0	0	1	1	1	1	1	0	1	0	IHM
c2	0	0	1	1	1	1	1	0	1	0	IHM
c5	0	0	0	1	0	1	1	0	0	0	IHM
m3	0	0	0	0	0	0	0	0	0	1	graphes
m4	0	0	0	0	0	0	0	0	1	1	graphes
m4	0	0	0	0	0	0	0	0	1	1	graphes

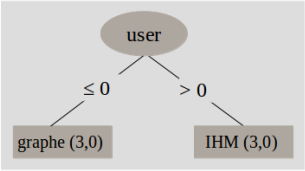


FIGURE 3.4 – Exemple de bootstrap ainsi que l'arbre généré sur la base des titres.

classe IHM, nous manquons ici de précision. Mais comme en réalité cet arbre va faire partie d'un ensemble d'arbres qui sont construits de la même manière, la probabilité d'erreur va baisser puisqu'un vote erroné va être compensé par les votes des autres classifieurs.

Avec l'apparition du Bagging, la notion de données "out of bag" est apparue. Il s'agit de l'ensemble des exemples qui ne sont pas choisis dans un bootstrap. Ces données sont généralement utilisées pour tester la performance du classifieur généré à partir du bootstrap en question.

3.4 Random Feature Selection

Le "Random Feature Selection" (RFS) est un deuxième niveau d'induction de classifieurs qui a été proposé par Breiman pour construire ses forêts aléatoires. Comme son nom l'indique, il ne s'agit plus de travailler dans cette méthode sur les données mais plutôt sur les caractéristiques. Cette idée est inspirée du travail de Amit et Geman dans [3] pour la reconnaissance d'écriture manuscrite. L'algorithme consiste à choisir *aléatoirement* et *sans remise* cette fois un ensemble de caractéristiques de taille $K < M$, où M représente la dimension de l'espace des caractéristiques, puis de construire le nœud de l'arbre en ne tenant compte que de cet ensemble. Puis, les critères de partitionnement traditionnels sont appliqués pour trouver la caractéristique la plus discriminante.

Cette méthode permet d'augmenter le niveau de l'aléatoire dans le processus d'induction des forêts, ce qui augmente directement la diversité au sein des arbres créés. C'est grâce à la quantité de l'aléatoire injectée par le Bagging et RFS que les forêts aléatoires ne souffrent plus du sur-apprentissage.

Le choix du paramètre K est important dans le processus de construction des forêts aléatoires. La valeur la plus utilisée est $K = \sqrt{M}$. Breiman a mené ses expérimentations en fixant K à 1 et à la partie entière de $log(M) + 1$. Bernard et al. dans [10] ont fait une étude complète sur le choix de ce paramètre. Ils ont proposé une nouvelle version des forêts aléatoires où K n'est plus un paramètre fixé par l'utilisateur mais il est défini selon une heuristique dépendant du gain d'information des caractéristiques de la base d'apprentissage.

Dans notre exemple de base de titres, à partir de la nouvelle base générée par le bagging nous appliquons la technique du RFS. Nous choisissons aléatoirement 3 caractéristiques ($\approx \sqrt{10}$) sans redondance et nous construisons de nouveau l'arbre,

en ne tenant compte que de ces caractéristiques. La figure 3.5 illustre ce processus.

titre	human	interface	computer	user	system	response	time	EPS	survey	minors	classe
c2	0	0	1	1	1	1	1	0	1	0	IHM
c2	0	0	1	1	1	1	1	0	1	0	IHM
c5	0	0	0	1	0	1	1	0	0	0	IHM
m3	0	0	0	0	0	0	0	0	0	1	graphes
m4	0	0	0	0	0	0	0	0	1	1	graphes
m4	0	0	0	0	0	0	0	0	1	1	graphes

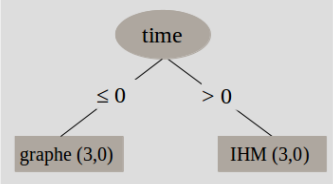


FIGURE 3.5 – Illustration du RFS sur la base de titres.

Le nouvel arbre généré fait partie de la forêt aléatoire à construire. Tout le mécanisme que nous venons de décrire est répété N fois, N étant le nombre d'arbres par forêt.

3.5 Force et corrélation

En proposant les forêts aléatoires, Breiman définit deux propriétés caractéristiques de cet algorithme d'apprentissage qui sont la force et la corrélation [15]. En utilisant ces deux propriétés, il est parvenu à montrer la convergence de son algorithme, ce qui revient à dire que les forêts aléatoires ne souffrent pas de sur apprentissage même si elles contiennent un grand nombre d'arbres.
La **force** est une propriété caractéristique de l'arbre qui représente sa fiabilité et son degré de performance au sein de la forêt. La **corrélation** est une caractéristique de l'ensemble total des arbres qui représente leur degré de dépendance mutuelle.
Mathématiquement ces deux propriétés sont définies comme suit. Soit la fonction marginale :

$$mg(X,Y) = av_k I(h_k(X) = Y) - \max_{j \neq Y} av_k I(h_k(X) = j) \tag{3.5}$$

où h_k et le k^{eme} classifieur dans l'ensemble et $I()$ est la fonction indicatrice qui vaut 1 si l'événement est vrai et 0 sinon. Cette fonction représente la différence entre la moyenne des votes des classifieurs pour la classe correcte et la moyenne des votes pour la classe erronée la plus prédite. Plus cette valeur est grande, plus les classifieurs prédisent la bonne classe. Dans le cas où elle est négative, la classe prédite n'est pas correcte.
A partir de la fonction marginale, l'erreur en généralisation est définie comme suit :

$$PE^* = P_{X,Y}(mg(X,Y) < 0) \tag{3.6}$$

C'est la probabilité que la forêt se trompe dans la classification. Dans le cas où l'ensemble des classifieurs est une forêt où $h_k(X) = h_k(X, \theta_k)$, Breiman a montré que la fonction marginale peut s'écrire :

$$mr(X,Y) = P_\theta(h(X,\theta) = Y) - \max_{j \neq Y} P_\theta(h(X,\theta) = j) \tag{3.7}$$

La force de la forêt est donc l'espérance mathématique de la fonction marginale

$$s = E_{(X,Y)} mr(X,Y) \tag{3.8}$$

En utilisant l'inégalité de Chebychev, l'erreur en généralisation sera :

$$PE^* \leq \frac{var(mr)}{s^2} \tag{3.9}$$

La variance de la fonction marginale peut être simplifiée comme suit : soit rmg la fonction marginale brute d'une forêt aléatoire définie comme suit :

$$rmg(\theta, X, Y) = I(h(X,\theta) = Y) - I(h(X,\theta) = \hat{j}(X,Y))) \tag{3.10}$$

où $\hat{j}$ (X,Y)) est l'indice de la classe erronée, $mr(X,Y)$ représente la valeur attendue du $rmg(\theta, X, Y)$ par rapport à θ.
Supposons que nous avons θ et θ' deux classifieurs indépendants et identiquement distribués.

$$mr(X,Y)^2 = E_{\theta,\theta'}[rmg(\theta, X, Y) rmg(\theta', X, Y)] \tag{3.11}$$

La variance est égale alors :

$$var(mr) = E_{\theta,\theta'}[\rho(\theta,\theta')\sigma(\theta)\sigma(\theta')] \tag{3.12}$$

où $\rho(\theta,\theta')$ est la corrélation entre $rmg(\theta, X, Y)$ et $rmg(\theta', X, Y)$. En exprimant la variance en fonction de la corrélation, Breiman a montré que l'erreur en généralisation peut être finalement définie comme suit :

$$PE^* \leq \frac{\bar{\rho}(1 - s^2)}{s^2} \tag{3.13}$$

où $\bar{\rho}$ représente la valeur moyenne de $\rho(\theta,\theta')$.

Comme nous pouvons le constater dans 3.13, l'erreur en généralisation est un consensus entre la force des arbres de la forêt et leur corrélation. Minimiser l'erreur en généralisation revient à augmenter la force et diminuer la corrélation. Les forêts aléatoires les plus performantes sont donc celles dont les arbres sont les plus efficaces individuellement et les moins corrélés.

3.6 Consistance des forêts aléatoires

D'après la thèse de CISS [23], la consistance est définie comme étant la capacité du classifieur à converger vers l'erreur de Bayes, soit l'erreur qui rend le classifieur optimal. Un classifieur consistant ne garantit pas une erreur de prédiction faible. Cependant, s'il est optimal au sens de l'erreur de Bayes, alors on ne peut pas faire mieux que ce classifieur. Pour deux classifieurs consistants, la mesure de leurs performances s'effectue en comparant leurs vitesses de convergence, c'est-à-dire, le nombre d'observations à partir desquelles le classifieur tend à devenir optimal. Du point de

vue théorique, la consistance est établie lorsque le nombre d'observations tend vers l'infini, et dans la pratique, lorsque ce nombre est assez important.

Biau et al. ont prouvé dans [11] la consistance des forêts aléatoires et ceci en montrant la consistance du *vote majoritaire* d'un ensemble de classifieurs quelque soit le classifieur de base utilisé. Ils ont aussi montré que l'introduction de *l'aléatoire* dans le processus de construction des forêts n'impacte pas leur consistance.

Soit la distribution (X, Y) avec une probabilité *a posteriori* $\eta : \mathbb{R}^d \to [0, 1]$

$$\eta(x) = \mathbb{P}\{Y = 1|X = x\}.$$

Où $(X, Y), (X_1, Y_2), ...(X_n, Y_n)$ est une base d'apprentissage nommée D_n et Y représente la classe de chaque donnée X. $Y \in \{1, 2, ...k\}$

Soit $g_n(X, D_n)$ un classifieur binaire de X dans D_n avec une probabilité d'erreur définie par :

$$L(g_n) = \mathbb{P}_{(X,Y)}\{g_n(X, D_n) \neq Y\}. \tag{3.14}$$

Le classifeur de Bayes qui minimise la probabilité d'erreur est dans ce cas :

$$g^*(x) = \{1 \ si \ \eta(x) > 1/2\} \tag{3.15}$$

Le risque d'erreur ou risque de Bayes est :

$$L^* = L(g^*) = \mathbb{P}\{g^*(x) \neq Y\} = min(\eta(x), 1 - \eta(x)) \tag{3.16}$$

Un classifieur aléatoire utilise une variable aléatoire Z pour calculer sa décision. Il s'agit donc d'une fonction arbitraire $g(X, Z, D_n)$ que Biau et al. ont abrévié à $g(X, Z)$. Sa probabilité d'erreur est alors :

$$L(g_n) = \mathbb{P}_{(X,Y),Z}\{g_n(X, Z, D_n) \neq Y\} \tag{3.17}$$

Soient $Z1, Z2, ..., Z_m$ des variables aléatoires indépendantes et identiquement distribuées, un vote majoritaire sur un ensemble de classifieurs aléatoires est représenté par :

$$g_n^{(m)}(x, Z^m, D_n) = \begin{cases} 1 \ if \ \frac{1}{m}\sum_{j=1}^{m} g_n(x, Z_j, D_n) \geq \frac{1}{2} \\ 0 \ sinon. \end{cases} \tag{3.18}$$

Pour qu'un ensemble de classifieurs avec un vote majoritaire soit consistant il faut que :

$$\mathbb{P}\{g_n^{(m)}(x, Z^m) = 0 \to 0\} \ quand \ \eta > \frac{1}{2} \tag{3.19}$$

$$\mathbb{P}g_n^{(m)}(x, Z^n) = \mathbb{P}\{1/m \sum_{j=1}^{m} i \ si \ g_n(x, Z_i) = 0 > 1/2\} \tag{3.20}$$

Par la loi des grands nombres :

$$lim_{(m\to\infty)} g_n^{(m)}(x, Z^m, D_n) = 1_{\{\mathbb{E}_{Zg_n}(x,Z) \geq 1/2\}} \tag{3.21}$$

alors :

$$\mathbb{P}g_n^{(m)}(x, Z^n) \leq 2\mathbb{E}[(1/m)\sum_{j=1}^{m} 1_{\{g_n(x,Z_i)=0\}}] \tag{3.22}$$

Par l'inégalitée de Marcov :

$$= 2\mathbb{P}\{g_n(x, Z) = 0\} \to 0. \tag{3.23}$$

3.7 Types de forêts aléatoires

Fort du succès des forêts aléatoires, plusieurs chercheurs ont adapté l'implémentation de base Forest_RI des forêts aléatoires en proposant de nouvelles variantes. Ces différentes versions représentent généralement une adaptation à un domaine particulier ou à la nature des données d'apprentissage. Simon Bernard dans sa thèse [8] fait l'état de l'art de ces implémentations, nous donnerons ici un aperçu global des méthodes les plus populaires.

Dans son article [15], Breiman propose les "forest_RC" (Random combination) qui sont utilisées dans le cas où la base d'apprentissage contient un nombre réduit de caractéristiques. Dans ce cas, les forêts traditionnelles ne sont pas très efficaces parce que le principe de diversité est diminué au sein du processus "Random Feature Selection" à cause du panel des caractéristiques qui n'est pas assez large. La solution proposée dans "forest_RC" consiste à construire de nouvelles caractéristiques en combinant les anciennes au lieu de se contenter d'un choix aléatoire à chaque nœud.

Deux autres types d'algorithmes ont été proposés en 2004 par Chen et al. dans [22] : "Balanced Random Forest (BRF)" et "Weighted Random Forest (WRF)". Le but de ces deux implémentations est d'améliorer les performances des forêts aléatoires dans le cas où les bases d'apprentissage sont mal équilibrées avec des classes sur et sous représentées. Il s'agit principalement de compenser la classe sous représentée sur laquelle le modèle est généralement mal appris et ceci pour diminuer l'erreur de prédiction de cette classe. Le principe des BRF change le choix totalement aléatoire des exemples au bagging en un choix aléatoire par classe : le même nombre d'exemples est trié pour chaque groupe. Un arbre est ainsi construit sur un ensemble de données équilibrées. Les "WRF" attribuent à chaque classe un poids dépendant de sa représentativité dans la base d'apprentissage. Une classe bien représentée est une classe qui a un poids inférieur à une classe mal représentée. Ces poids sont pris en compte lors du choix de la meilleure caractéristique pour la construction du nœud et lors de l'agrégation des votes qui n'est plus majoritaire mais plutôt pondéré.

En 2006, Geurts et al. dans [32] proposent une autre variante des forêts aléatoires qu'ils appellent "Extremely Randomized Trees" ou "Extra trees" dans lesquelles la diversité est augmentée en injectant encore de l'aléatoire dans leur processus de construction. Le principe du bagging est supprimé et un partitionnement complètement aléatoire des données remplace l'utilisation d'un critère de partitionnement à chaque nœud. L'intérêt de la méthode est sa capacité à offrir un meilleur compromis entre le couple biais/variance, mais elle reste très complexe à interpréter car composée d'arbres de grande taille.

3.8 Forêts aléatoires et apprentissage dynamique

Les différents types de forêts aléatoires dont nous avons parlé dans la section 3.7 font partie des algorithmes d'apprentissage statiques : les forêts sont construites en une unique itération. En effet, l'algorithme d'apprentissage prend en entrée la totalité de la base d'apprentissage et il fournit en sortie un modèle qui reste inchangeable tout au long de la classification. Il existe un autre type d'apprentissage pour lequel la construction du modèle de classification n'est pas faite en une seule itération : le modèle se met en effet à jour d'une façon répétitive. Il s'agit de l'apprentissage dynamique. Ce type d'apprentissage est généralement utilisé soit pour améliorer l'approche statique avec de nouvelles données soit pour répondre à une exigence particulière dans laquelle il n'est pas possible d'appliquer l'apprentissage statique, comme par exemple la non disponibilité de la base d'apprentissage en amont de la phase d'apprentissage. Plusieurs variétés d'arbres de décisions et de forêts aléatoires construits d'une façon dynamique existent dans la littérature. Bernard et al. dans [9] proposent les forêts aléatoires dynamiques ("Dynamic Random Forest"). En suivant le principe du boosting [62], qui est un autre algorithme d'induction d'un ensemble de classifieurs, les forêts dynamiques consistent à construire d'une façon itérative un ensemble d'arbres de décision où le $i^{ème}$ arbre compense l'erreur de ceux de l'itération $i-1$. Il est donc nécessaire de définir un critère d'évaluation des différents arbres construits en attribuant un score de confiance à chaque donnée d'apprentissage. Ce score se base sur le nombre d'arbres qui ont prédit correctement cette donnée. La première itération de l'algorithme consiste à attribuer des poids aux données d'apprentissage d'une façon équiprobable. Puis, les poids sont actualisés d'une itération à une autre de telle façon que si une donnée est mal classée par un nombre important d'arbres, alors elle est favorisée dans la prochaine étape. L'algorithme peut ainsi mieux apprendre sur cette donnée. Le poids d'une donnée est alors inversement proportionnel à son score de confiance. Bernard et al. ont montré qu'une telle construction permet de fournir des arbres qui maximisent leurs forces et minimisent leur corrélation. Ils ont montré aussi que les résultats obtenus sont intéressants si on les compare avec les forêts aléatoires traditionnelles.

Un autre type d'apprentissage dynamique est l'apprentissage en ligne. Ce type est conçu pour répondre à la problématique des données qui sont construites d'une façon séquentielle (en "stream"). Elles sont généralement de grande taille car elles arrivent d'une façon fréquente au cours du temps et en paquets. Abdulsalam et al. présentent dans [1] les différents scénarios possibles de distribution des données d'apprentissage (figure 3.6).

Les données d'apprentissage dans le scénario 0 arrivent toutes au début du processus. L'apprentissage dans ce cas est similaire à un apprentissage statique mais le modèle doit être construit rapidement pour classer les nouvelles données. Dans le scénario 1, les données arrivent d'une façon régulière au cours du temps mais elles représentent des gros paquets, un algorithme peut être crée et testé à partir du premier paquet mais il doit s'adapter aux nouveaux paquets qui arrivent. Dans le scénario 2 les données arrivent d'une façon régulière aussi mais avec de petites

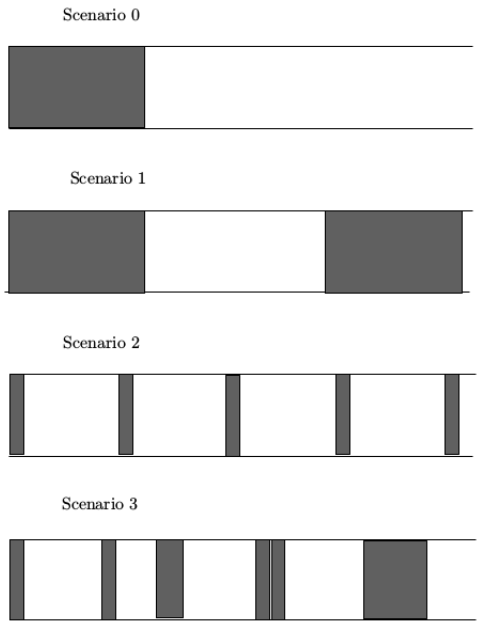


FIGURE 3.6 – Les scénarios de distribution des données d'apprentissage [1].

quantités qui ne suffisent pas pour construire un modèle consistant dès les premières itérations. La classification peut commencer très tôt dans ce cas, mais puisque la fréquence des données est connue, l'algorithme est averti et la classification peut être retardée jusqu'à la construction d'un modèle plus robuste. Le dernier scénario présente des données qui arrivent au cours du temps en paquets non égaux, de petite taille et de façon non régulière. Ce scénario est celui qui est le plus difficile mais qui correspond également le plus au cas réel. Ce type de scénario a besoin parfois de l'intervention de l'utilisateur pour construire le modèle. Ainsi si l'on souhaite détecter si un message électronique est un "spam" ou non, l'algorithme peut se référer aux actions de l'utilisateur en considérant le message comme étant spam s'il a été supprimé sans être lu.

Pour résumer, l'apprentissage en ligne nécessite de façon générale une construction incrémentale du modèle de classification. Plusieurs défis se posent pour la construction d'un tel modèle. Nous les résumons dans les points suivants :

— il doit être proche de celui construit statiquement ;
— il doit fournir des résultats de classification à n'importe quel moment de sa construction avec un haut niveau de performance ;
— il doit s'adapter facilement aux nouvelles données ;
— il doit être rapide avec un temps limité d'apprentissage et de classification.
— il ne doit pas consommer beaucoup de mémoire.

Pour répondre à ces défis, plusieurs constructions en ligne des arbres de décisions et des forêts aléatoires sont proposées. En 2000, Domingos et Hulten dans [28] ont proposé les arbres de Hoeffding. Le but est ici de créer un arbre de décision d'une façon incrémentale à partir d'une large base potentiellement infinie de données.

Chaque exemple dans la base doit être traité en une seule fois et dans un temps constant car le modèle doit assurer une classification en temps réel. Il est montré dans [20] que pour construire un nœud de l'arbre il suffit de considérer un ensemble réduit de données. Après un certain nombre d'exemples, quand le gain d'information du meilleur attribut dépasse le deuxième meilleur attribut d'un seuil donné, le nœud courant est décomposé et les exemples sont transférés à ses feuilles. Le calcul du bon moment de partitionnement du nœud se base sur un résultat statistique qui est l'inégalité de Hoeffding [51] [34] qui sera détaillée dans le chapitre 5. De cette façon, Domingos et Hulten montrent que le meilleur attribut calculé à partir d'un petit ensemble de données est le même que celui calculé à partir d'un ensemble beaucoup plus large.

Dans le même contexte, Pfahringer et al. proposent dans [58] un nouvel algorithme pour la fouille des données en "stream". Leur algorithme représente une extension des arbres de Hoeffding qui est conçu pour pallier les limites de ces arbres ainsi que celles des ensembles de classifieurs. En effet, selon Pfahringer et al. le score d'évaluation d'un nouvel exemple à classer dépend d'un seul chemin dans l'arbre de décision d'une façon plus générale. De plus, le fait qu'un nœud ne soit découpé que par un seul attribut créé une sorte d'instabilité dans les arbres. En effet, les autres attributs peuvent avoir une performance très proche de celui qui a été utilisé (le meilleur), il peuvent même mener à d'autres arbres assurant une meilleure classification. D'un autre côté, Pfahringer et al. ont démontré les limites des algorithmes d'ensemble de classifieurs qui fournissent des modèles non interprétables et de grandes tailles. Dans le mode statique, une solution proposée pour contourner ces limites est les arbres d'options [17] [45] : il s'agit de regrouper plusieurs arbres dans une seule structure où des nœuds options sont introduits afin qu'un exemple traverse plusieurs schémas à la fois. La figure 3.7 présente un arbre d'options. A la fin de la classification, un exemple se retrouve dans plusieurs feuilles, un vote majoritaire permet de prendre la décision finale.

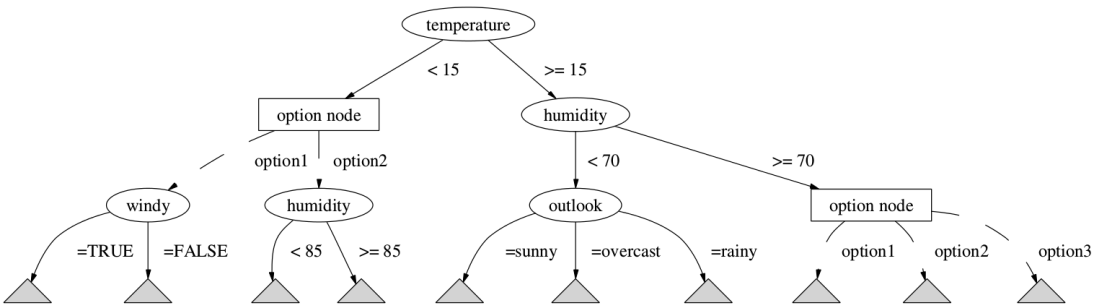


FIGURE 3.7 – Exemple d'un arbre d'options [58].

Pfahringer et al. dans [58] combinent les arbres de Hoeffding avec les arbres d'options pour obtenir un nouvel algorithme d'apprentissage itérable et peu gourmand en mémoire. Pour cela, un mécanisme de gestion d'agrandissement de l'arbre est

mis en place : l'idée est d'utiliser l'algorithme proposé par Domingos et Hulten dans [28] en lui ajoutant le calcul nécessaire pour déterminer le meilleur moment pour ajouter le nœud option dans l'arbre. Une nouvelle option est ajoutée dans l'arbre si le meilleur attribut dans l'ensemble des attributs non utilisés est meilleur que celui répartissant le nœud option courant en terme de gain d'information et de l'inégalité de Hoeffding. Un vote pondéré permet de prendre la décision finale. Pour contrôler l'agrandissement de l'arbre, un ensemble de paramètres est mis en place comme par exemple le nombre de nœuds options maximal par nœud ou le nombre maximum de chemins possibles. Tout ce processus est accompagné par une procédure d'élagage qui consiste à supprimer les parties redondantes ainsi que les parties impactant négativement l'accuracy.
Pfahringer et al. ont testé leur méthode sur une variété de benchmarks. Les résultats obtenus pour l'accuracy ont montré que les arbres de Hoeffding avec les options sont meilleurs en moyenne que les arbres de Hoeffding classiques [28].

Nous avons vu jusqu'à maintenant comment construire un arbre de décision en ligne, plusieurs travaux existent dans la littérature pour construire une forêt aléatoire en ligne également. Oza dans [55] a proposé le bagging en ligne. Nous rappelons le principe du bagging [14] qui consiste à choisir aléatoirement et avec remise un sous ensemble d'exemples de la base d'apprentissage de taille N. Ce sous ensemble sert à construire un premier classifieur. Chaque sous ensemble tiré contient k copies des exemples le composant selon une distribution binomiale :

$$P(K = k) = \binom{N}{k} \left(\frac{1}{N}\right)^k \left(1 - \frac{1}{N}\right)^{(N-k)} \tag{3.24}$$

En apprentissage en ligne le nombre d'exemples constituant la base d'apprentissage est infini. Quand N tend vers l'infini la distribution de k tend alors vers Poisson(1) :

$$P(K = k) \sim exp(-1)/k! \tag{3.25}$$

Le bagging en ligne opère alors comme suit : quand un nouvel exemple d'apprentissage est disponible, on génère aléatoirement et suivant la loi de Poisson un nombre $k \sim Poisson(1)$, puis, on entraîne le modèle avec cet exemple k fois. De cette façon, le bagging en ligne représente une bonne approximation du bagging en mode statique. Les modèles fournis par ces deux algorithmes sont similaires puisqu'ils sont appris sur des distributions similaires. Les expérimentations ont également montré que les deux méthodes convergent vers le même classifieur pour la même base d'apprentissage quand le nombre de modèles et les exemples d'apprentissage tendent vers l'infini.

L'implémentation du bagging en ligne proposée par Oza a été utilisée dans les travaux de Saffari et al. [66]. Ces travaux portent sur la construction d'une forêt aléatoire en ligne. Les auteurs ont soulevé les mêmes problèmes que lors d'une construction itérative de la forêt. En effet, une fois qu'un nœud d'un arbre est construit, il est impossible de le modifier ou de le corriger par les nouvelles données. Il faut déterminer dans ce cas le bon moment de partitionnement tout en proposant

des solutions qui ne consomment pas beaucoup de mémoire. A l'image des forêts aléatoires, l'idée de l'algorithme est de construire un bootstrap en ligne pour chaque nouvel arbre. Lorsqu'un nouvel exemple se présente, k, le nombre de copies de cet exemple, est généré selon la loi de Poisson. Si $k = 0$ cet exemple ne fera pas partie de ce bootstrap mais il est utilisé comme donnée "Out Of Bag" (OOB) pour tester l'arbre. Si $k > 0$ alors l'exemple est ajouté k fois dans l'ensemble d'apprentissage. A chaque nouvel ajout deux conditions sont vérifiées pour partitionner le nœud courant : le nombre d'exemples dans le bootstrap doit dépasser un certain seuil α et un minimum de gain d'information est exigé pour répartir le nœud en dépassant un autre seuil β. Une fois ces deux conditions vérifiées, le nœud est réparti en deux fils gauche et droit. La distribution des données va être propagée à ces deux nœuds fils. Ce traitement va être répété jusqu'à la convergence de l'arbre. Le principe d'agrandissement de la forêt proposé est inspiré des Etrees, ou "Evolving trees", [56] qui consistent à compter le nombre d'observations pour chaque nœud et à le décomposer à partir d'un certain seuil d'observations. Saffari et al. ont affirmé que les conditions de partitionnent d'un nœud dans les arbres de Hoeffding ou dans les ETrees sont bien applicables dans leur implémentation, mais ils préfèrent quand même leur approche car elle calcule continuellement le gain d'information des caractéristiques permettant ainsi de mieux apprendre le modèle, ce qui convient mieux à la nature inhérente des arbres de décision. Pour assurer l'adaptabilité au cours du temps et comme dans certaines applications les données peuvent changer, les arbres créés sont testés par les données OOB, un certain nombre d'entre eux est supprimé selon leur erreur de out of bag. Les expérimentations menées sur les forêts en ligne ont montré des résultats prometteurs dans le sens où ce modèle finit par converger vers celui construit en mode statique. En les comparant avec le boosting en ligne [33], les forêt en ligne prennent l'avantage car elles sont plus robustes au bruit.

Dans le même contexte, une autre extension de forêts aléatoires est proposée dans le cadre de l'apprentissage en ligne. Il s'agit du "Streaming Random Forest" [1]. La condition de partitionnement d'un nœud dans cet algorithme dépend à la fois d'un ensemble de paramètres et de l'inégalité de Hoeffding proposée dans [28]. L'algorithme consiste alors à définir initialement le nombre total d'exemples à utiliser dans chaque arbre (*tree window*), puis pour construire chaque nœud un comptage du nombre d'exemples est effectué. Quand ce nombre dépasse un minimum donné (n_{min}), la condition de mise à jour est vérifiée par rapport à l'inégalé Hoeffding. Si après un certain nombre d'itérations, la condition basée sur l'inégalité Hoeffding n'est toujours pas vérifiée et qu'un nombre maximal d'exemples par nœud est atteint (*node window*), alors l'algorithme vérifie s'il existe une classe prédominante dans cet ensemble, dans ce cas le nœud est transformé en feuille contenant la classe prédominante. Dans le cas contraire, le nœud est décomposé par le meilleur attribut. Le *node window* dépend de la profondeur du nœud dans l'arbre, il est défini comme suit :

$$\frac{1}{\alpha}(treewindow/2^{profondeurDuNoeud}) \tag{3.26}$$

où α est un paramètre fixé empiriquement et *node window* diminue linéairement

en augmentant la profondeur car le nombre d'exemples est de plus en plus petit en descendant dans l'arbre. Une procédure d'élagage est nécessaire pour le "Streaming Random Forest" pour supprimer les nœuds qui ne contiennent pas un nombre suffisant d'exemples quand le *tree window* est atteint. L'élagage est également nécessaire quand un nœud père donne naissance à deux fils qui n'ont pas réussi à recevoir un nombre suffisant d'exemples. La pureté de chaque nœud fils est ainsi calculée : elle est définie comme étant le ratio entre le nombre d'exemples de la classe majoritaire et le nombre total des exemples dans ce nœud. Si la pureté des deux fils est inférieur à $\frac{1}{nombre\ de\ classes}$ alors ils sont supprimés et leur père est considéré comme une feuille à qui la classe majoritaire est attribuée. Sinon, les nœuds fils se transforment en feuilles avec la classe majoritaire de chacun. Les "Streaming Random Forest" ont été testées avec deux types de base données : synthétique et réelle. Elles ont été comparées avec les forêts traditionnelles. Le taux d'erreur de classification montre des résultats similaires pour les deux algorithmes, ce qui confirme l'efficacité de la méthode proposée et sa capacité à produire un modèle qui converge vers les forêts traditionnelles.

Nous venons de citer les algorithmes des arbres de décision et des forêts aléatoires en ligne les plus populaires dans la littérature. Dans le même contexte, une plate-forme open source est disponible en ligne pour la classification des données en "stream" MOA [12]. Son but est d'analyser, de comparer et d'évaluer les différentes approches dans ce domaine à travers la création et l'utilisation de plusieurs benchmarks. Elle fournit donc différents générateurs de flux de données tels que "SEA Concepts Generator" [75] et "Function Generator" [2]. Elle fournit également un ensemble d'algorithmes d'apprentissage en ligne tels que les arbres de Hoeffding [28] et les arbres de Hoeffding options [58]. Enfin, la plate-forme offre des méthodes d'évaluation en ligne. Cette tâche pose également de nouveaux défis par rapport à l'évaluation en mode statique car l'ensemble des données n'est pas défini à l'avance. Comment est-il possible de créer une image de la métrique d'évaluation tout au long du processus d'apprentissage et de classification ? Pour répondre à cette question MOA propose deux approches :

- *Holdout* : définir un ensemble de données dans le "stream" comme un ensemble de tests et faire le test d'un seul coup.
- *Interleaved Test Then Train or Prequential* [30] : chaque nouvel exemple d'apprentissage est utilisé pour tester la performance du modèle avant d'être utilisé pour l'apprentissage.

L'architecture de MOA fait que ces différentes étapes sont extensibles. Il est donc facile d'y ajouter de nouveaux algorithmes de génération de données, d'apprentissage ou d'évaluation de modèle en ligne.

3.9 Synthèse

Nous avons fait dans ce chapitre un rappel sur les forêts aléatoires [15] qui sont un ensemble d'arbres de décision où chaque arbre vote pour une classe et où la décision

finale est prise selon un vote majoritaire. Les forêts aléatoires les plus efficaces sont celles qui proposent les arbres les plus performants individuellement et les moins corrélés entre eux. La diversité introduite par le processus aléatoire caractérisant la construction d'un tel algorithme représente une clé de réussite de cette méthode. Cependant, dans certains cas, il est nécessaire d'adapter cet algorithme pour mieux répondre à des besoins particuliers et pour éviter quelque limites de la méthode. Ainsi, plusieurs implémentations dérivées des forêts aléatoires ont été proposées dans la littérature comme par exemple les Balanced Random Forest, les Random Forest Combinaison, ou encore les forêts en ligne.

Nous adoptons pour la suite cet algorithme d'apprentissage comme base de nos travaux sur la classification des textes courts. Plusieurs raisons nous ont mené à ce choix. La popularité et les performances remarquables de cette méthode dans différents domaines notamment la classification des textes comme dans [44] en font partie. De tels résultats peuvent être anticipés dans le domaine des messages courts. En effet, le processus de Random Feature Selection appliqué sur un espace large de caractéristiques issues des textes courts permet d'atteindre un grand niveau de diversité, ce qui constitue un facteur important pour le succès des forêts aléatoires.

Par ailleurs, techniquement, les forêts aléatoires sont rapides en terme d'apprentissage et de classification puisqu'elles sont facilement parallélisables. Elles présentent aussi un avantage par rapport aux autres algorithmes de la famille ensemble de classifieurs puisque les arbres de décisions supportent le multi classes.

La nature de construction des forêts aléatoires est également très adaptée à nos travaux. En effet, Le mécanisme de sélection de caractéristiques intégré dans la méthode fait qu'elle se base sur une segmentation sémantique des données ce qui peut nous aider dans l'idée de l'introduction de la sémantique dans un algorithme d'apprentissage qui est un des objectifs de cette thèse.

Enfin, du point de vue interactivité, qui est un autre objectif de cette thèse, un mécanisme de mise à jour peut être implémenté dans les forêts aléatoires en s'inspirant de la littérature de l'apprentissage en ligne que nous avons présenté dans ce manuscrit.

Dans le prochain chapitre, nous allons présenter notre première contribution qui est les forêts sémantiques, une nouvelle approche pour la classification des textes courts se basant sur la sémantique.

Chapitre 4

Forêts Sémantiques

Sommaire

4.1 Introduction **63**
4.2 Enrichissement **65**
4.2.1 Enrichissement niveau mot 68
4.2.2 Enrichissement niveau message 68
4.3 Introduction de la sémantique dans les forêts aléatoires . . **71**
4.4 Expérimentations et résultats **74**
4.4.1 Benchmark 74
4.4.2 Résultats et interprétations 75
4.5 Tests statistiques **81**
4.6 Synthèse **85**

4.1 Introduction

Comme nous l'avons déjà mentionné précédemment, la nature des messages courts rend leur classification plus difficile que celle des textes traditionnels. Nous avons proposé dans le chapitre 2 l'état de l'art du processus complet de la classification de ces messages. Afin d'améliorer ce processus, la plupart des recherches se concentrent sur l'amélioration de la qualité des textes par le biais de la sémantique. Nous avons identifié deux grands types de travaux d'amélioration, le premier consiste à étendre les textes en utilisant les ontologies ou les techniques du Topic Models par exemple. Le but dans ce cas est d'apporter de la connaissance externe aux messages permettant ainsi de mieux identifier leurs contextes et d'augmenter leur densité. Le deuxième type attaque le problème d'un point de vue complètement différent, une autre communauté travaille plutôt sur la réduction de l'espace des caractéristiques des messages, ce qui permet de construire une matrice plus compacte au sein de la quelle l'identification des liens sémantiques entre les mots est plus facile, comme cela a été démontré dans [88]. Plusieurs techniques ont été exploitées à cet effet telles que l'abstraction et la sélection des caractéristiques. Nous remarquons ici que tous ces travaux sont effectués au niveau de la phase de prétraitement de la chaîne complète du processus de classification. En effet, une fois la qualité des messages est améliorée, les algorithmes d'apprentissage traditionnels sont utilisés pour construire le modèle de classification. Nous estimons qu'un algorithme d'apprentissage spécifique aux messages courts peut améliorer d'avantage leur classification en

tenant compte des liens sémantiques existants, une phase de pré apprentissage étant toujours nécessaire pour améliorer la qualité des messages.

Nous avons présenté dans le chapitre 3 les forêts aléatoires [15] : un algorithme d'apprentissage démontré puissant et efficace dans plusieurs domaines dont la fouille des textes. Il s'agit en effet d'un algorithme consistant, qui converge vite et dont la décision finale de classification n'est pas prise par un seul classifieur mais par plusieurs. Les forêts aléatoires sont facilement extensibles et modulables. Plusieurs types de forêts sont inférés à partir de l'implémentation de base pour répondre à des exigences particulières. D'un autre côté, la nature de construction des forêts se basant sur une segmentation sémantique des données est parfaitement adaptée avec l'idée de l'algorithme que nous cherchons à implémenter.

Dans ce chapitre nous proposons une nouvelle approche pour la classification des messages courts. Cette approche se base principalement sur l'introduction de la sémantique dans les différentes étapes du processus de la classification supervisée. Elle consiste, en premier lieu, à proposer une nouvelle méthode d'enrichissement pour améliorer la qualité des textes. Elle met en place, en deuxième lieu, un nouvel algorithme d'apprentissage qui tient compte de la sémantique des messages dans son comportement. Comme elle se base principalement sur les forêts aléatoires, nous appelons notre approche "les forêts sémantiques" (FS). La figure 4.1 donne un aperçu général sur la méthode proposée.

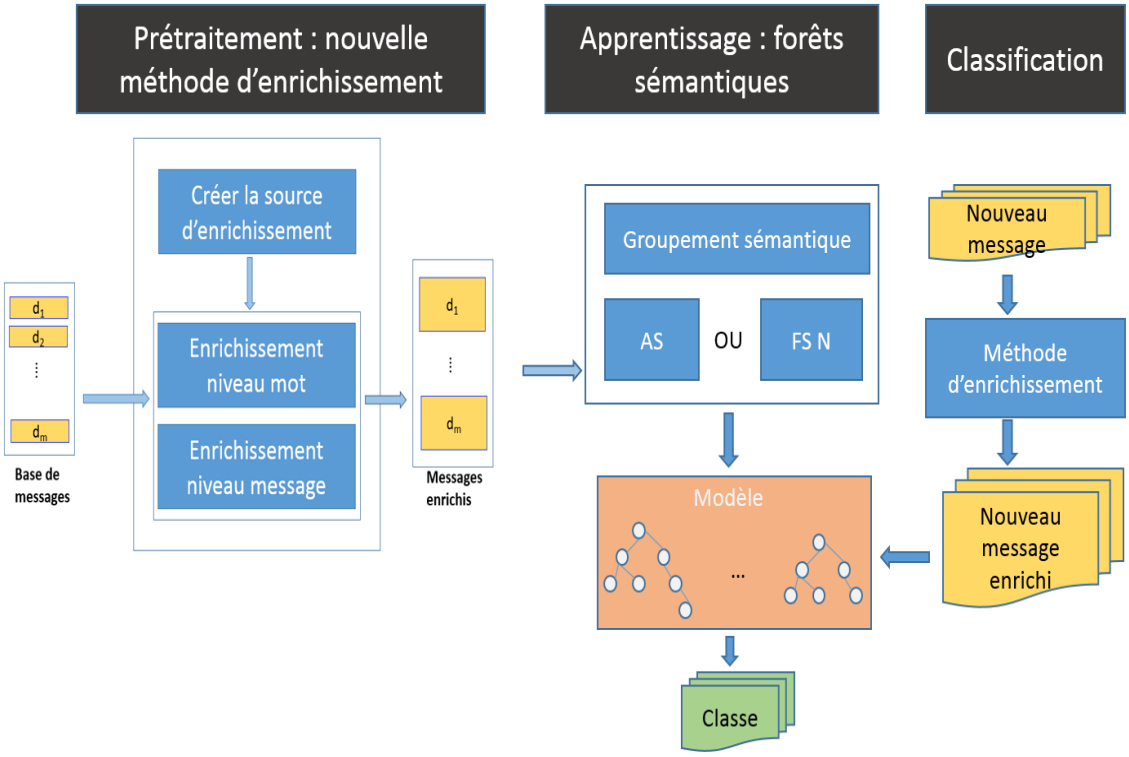


FIGURE 4.1 – Schéma général des forêts sémantiques.

La première étape des forêts sémantiques consiste à enrichir les messages de la base d'apprentissage un à un par de la connaissance externe. Une phase préliminaire de pré enrichissement consiste à créer une base de connaissance externe puis de l'utiliser pour un enrichissement sur deux niveaux : niveau mot et niveau message. Une fois la nouvelle base de messages enrichis obtenue, nous disposons alors d'un espace de caractéristiques (espace de mots) bien plus large que celui avant enrichissement. L'étape suivante consiste à appliquer un nouveau type de forêts aléatoires dont le comportement se base aussi sur la sémantique des données. Deux versions de forêts sémantiques sont proposées : les forêts sémantiques au niveau arbres ou les arbres sémantiques (AS) et les forêts sémantiques au niveau nœuds (FS N). Le processus d'induction de ces deux implémentations nécessite un groupement sémantique de l'espace de caractéristiques afin d'établir des connexions sémantiques entre ces éléments. La construction d'une forêt dépendra ensuite de ces connexions. Une fois que le modèle de classification est créé, quand un nouveau message arrive, il est enrichi avec le même mécanisme puis classé par ce modèle.

Dans la suite de ce chapitre la nouvelle méthode d'enrichissement ainsi que le nouvel algorithme d'apprentissage dans ses deux versions seront expliqués en détail. Nous présentons aussi les différentes expérimentations menées sur cette nouvelle approche ainsi que leurs résultats.

4.2 Enrichissement

Le but de cette étape est de pallier les limites des messages courts que nous avons cités dans le chapitre 2, à savoir la parcimonie et le manque de contexte. L'idée générale de notre méthode est de transformer un message court composé d'un petit ensemble de mots en un plus grand. Afin de mieux identifier le contexte des messages, l'information ajoutée doit être sémantiquement proche du texte d'origine. Nous avons besoin ici d'une source d'enrichissement qui apporte la connaissance externe à nos messages. Cette source doit être à la fois spécifique à la base d'apprentissage, pour ne pas introduire du bruit, et complète, pour couvrir tous les domaines de cette base. Les ontologies universelles comme WordNet et Mesh sont de bonnes sources d'enrichissement vu qu'elles sont riches en information mais elles restent trop générales et non spécifiques à un domaine particulier. Les dictionnaires sont aussi une source d'enrichissement importante mais ils ne sont pas adéquats à ce que nous cherchons vu qu'ils fournissent seulement des synonymes alors que nous cherchons la sémantique générale du message. Le mécanisme de création de la base d'enrichissement introduit par Phan dans [59] semble la meilleure solution puisqu'il rend possible de créer une source par domaine, ce qui permet d'offrir un enrichissement plus précis et plus complet surtout pour les domaines qui ne sont pas d'ordre général. Nous adoptons par la suite cette technique.
Pour rappel, la façon dont Phan créé sa source d'enrichissement est la suivante : il applique LDA [13], qui est une technique du "topic model", sur un ensemble de documents externes sémantiquement proches de la base d'apprentissage. LDA étant

un modèle probabiliste génératif qui consiste à extraire à partir d'une collection de documents un ensemble de thèmes cachés. Un thème est un ensemble de mots. Chaque mot possède un poids indiquant son importance dans le thème. L'ensemble des thèmes obtenus représente notre base d'enrichissement.

Une fois les thèmes construits, l'étape suivante consiste à les intégrer dans les messages. Plusieurs mécanismes d'enrichissement de messages existent dans la littérature mais nous avons remarqué que ces travaux considèrent le message comme étant un ensemble de mots où chaque mot est enrichi séparément. A notre connaissance, il n'existe pas de travaux qui considèrent un message comme une seule entité et l'enrichissent selon son sens général. Tenir compte du sens général du texte permet de résoudre quelques cas d'ambiguïté sémantique notamment la polysémie. Il permet aussi de mieux identifier le contexte. Dans cette thèse nous proposons une nouvelle méthode d'enrichissement sur deux niveaux : le niveau mot et le niveau message. Elle est résumée dans la figure : 4.2.

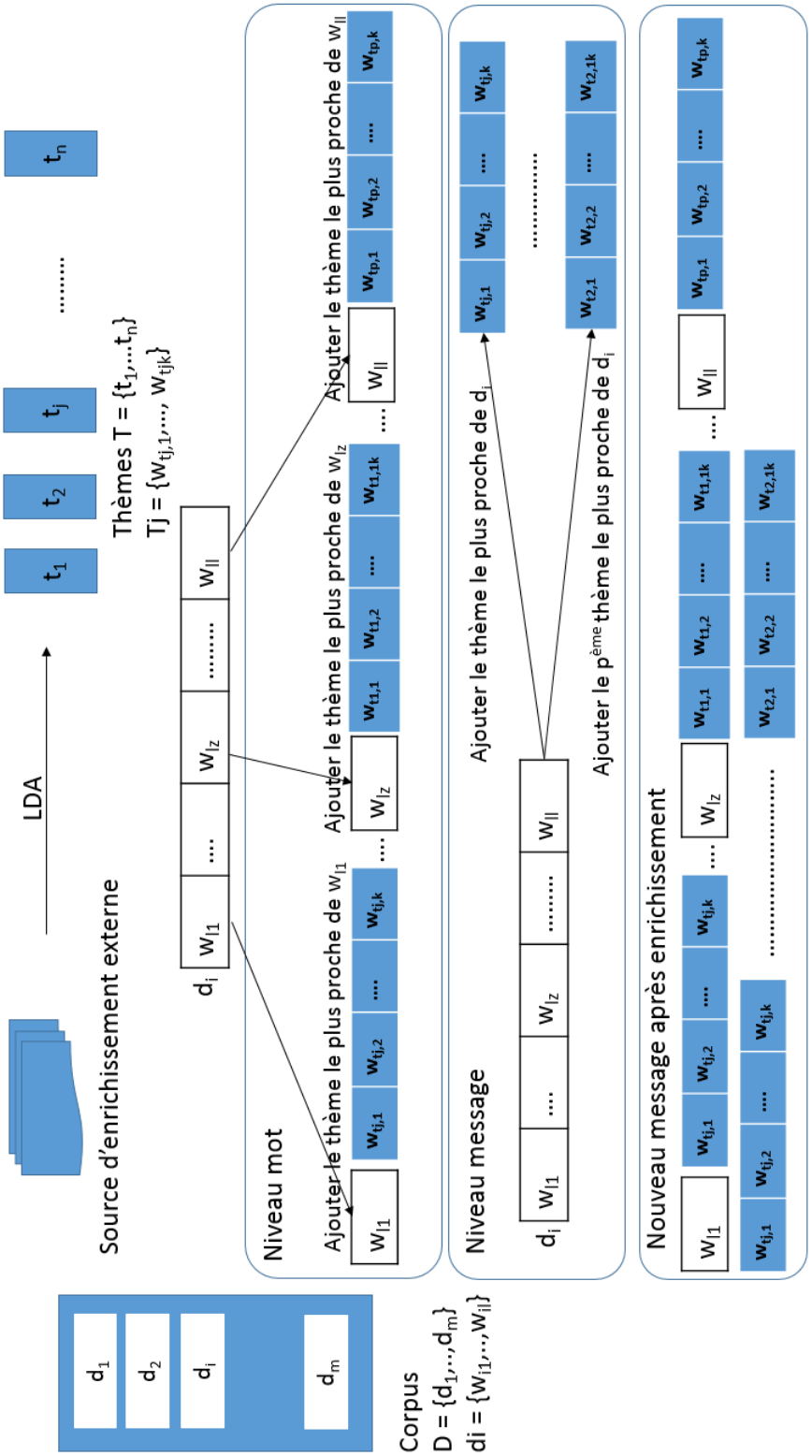


FIGURE 4.2 – Algorithme d'enrichissement.

4.2.1 Enrichissement niveau mot

L'enrichissement au niveau mot est un enrichissement local qui consiste à chercher pour chaque mot du message le thème le plus proche de ce mot, puis d'ajouter au vecteur représentatif du message tous les mots du thème identifié.

Soit $D = \{d_1, .., d_m\}$ une base de m messages à enrichir ; d_i est donc un message appartenant à la base D. Il est représenté par un ensemble de mots $d_i = \{w_{i1}, ..., w_{il}\}$. Soit $T = \{t_1, .., t_n\}$ une base de n thèmes et po^i_{zj} le poids du mot w_{iz} dans le thème t_j avec $z \in \{1, .., l\}$. La similarité entre w_{iz} et t_j est donnée par po^i_{zj} si w_{iz} appartient à t_j et elle est égale à 0 si le mot n'appartient pas au thème :

$$\text{sim}(w_{iz}, t_j) = \begin{cases} po^i_{zj}, & \text{si } w_{iz} \in t_j; \\ 0, & \text{autrement.} \end{cases} \tag{4.1}$$

Le thème t^* dans lequel le mot w_{iz} a le poids le plus grand est donné par :

$$t^* = \arg\max_j(\text{sim}(w_{iz}, t_j)). \tag{4.2}$$

L'enrichissement au niveau mots permet d'apporter de l'information massive et sémantiquement liée aux messages courts. En effet, il transforme un message composé de seulement une dizaine de mots en un message beaucoup plus conséquent composé d'un ensemble de concepts généraux. Puisqu'un mot n'est plus représenté par une entité singulière mais plutôt par un thème. Ceci permet de créer un modèle générique, riche et capable de classer n'importe quel message lié aux domaines de l'ensemble de données de départ même si les mots de ce message ne font pas partie du vocabulaire de l'ensemble d'apprentissage initial.

4.2.2 Enrichissement niveau message

Ce deuxième niveau représente un enrichissement global. Il vise à comprendre le sens général du texte court en considérant tous ces mots comme une seule entité. L'idée est donc d'enrichir le message par les p thèmes qui lui sont les plus proches sémantiquement. Nous définissons alors l'occurrence d'un thème dans un texte. Cette mesure se base sur le nombre de mots en commun entre un texte et un thème. L'occurrence d'un mot w_{iz} dans un thème t_j, $\text{occ}(w_{iz}, t_j)$ est donnée par :

$$\text{occ}(w_{iz}, t_j) = \begin{cases} 1, & \text{si } w_{iz} \in t_j; \\ 0, & \text{autrement.} \end{cases} \tag{4.3}$$

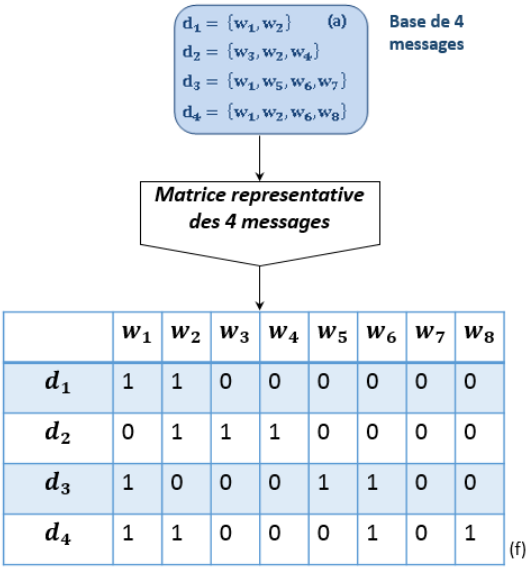
Le thème t^{max} le plus proche du texte est celui qui a l'occurrence la plus grande.

$$t^{max} = \arg\max_j(\text{occ}(\{w_{i1}, \ldots, w_{il}\}, t_j)). \tag{4.4}$$

Une fois les occurrences de tous les thèmes calculées, Elles sont triées puis tous les mots composant les p premiers thèmes sont ajoutés au message.
Comme l'enrichissement au niveau mot, ce deuxième niveau permet également

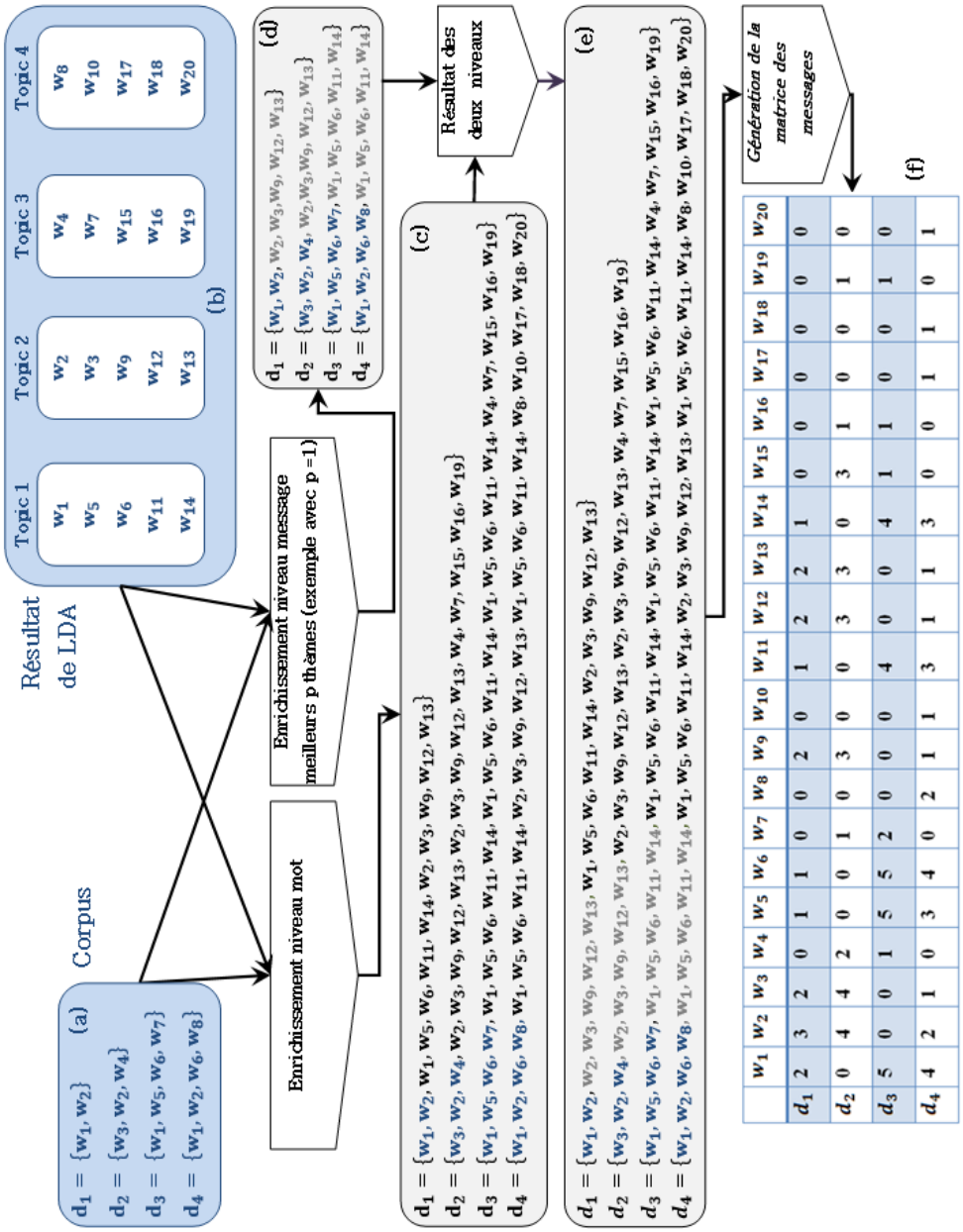
d'élargir le vocabulaire de la base initiale par la connaissance apportée de l'extérieur. Il résout aussi certains cas d'ambiguïtés sémantiques telle que la polysémie. Nous prenons ici par exemple le terme "jumelle" qui a deux significations : soit deux sœurs, soit l'instrument optique. En prenant le mot tout seul, nous n'arrivons pas à identifier le sens voulu de ce terme, par contre, en tenant compte de tous les mots autour, nous avons plus de chance de comprendre le contexte général et donc son vrai sens. L'enrichissement dans ce niveau est donc plus fin. En plus de ce problème de polysémie, ce processus permet aussi de résoudre le cas d'enrichissement d'un terme dont le sens est général et qui peut donc appartenir à plusieurs thèmes. Prenons par exemple le mot "compétition". Ce terme peut appartenir à la thématique sport s'il s'agit d'une compétition sportive ou à la thématique art s'il s'agit d'une compétition artistique. En cherchant le thème ayant le plus de mots en commun avec un message de départ contenant le mot "compétition", nous cherchons alors le thème indiquant le vrai sens du terme "compétition" dans le contexte du message.

En plus de l'élargissement du vocabulaire de la base d'apprentissage, le mécanisme d'enrichissement proposé permet d'améliorer la qualité des messages en diminuant leur parcimonie. En effet, comme le texte devient plus long après les deux phases d'enrichissement, le vecteur représentatif de ce texte est donc moins vide car il devient en effet plus grand et contient moins de valeurs nulles. Les figures 4.3 et, 4.4 montrent une illustration d'un exemple de base de messages avant et après le mécanisme d'enrichissement proposé. Nous pouvons remarquer la différence de la densité des deux matrices générées en termes de taille et de valeurs non nulles. Dans le chapitre 6, nous étudierons la variation du niveau de la parcimonie avant et après cette méthode d'enrichissement.



	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
d_1	1	1	0	0	0	0	0	0
d_2	0	1	1	1	0	0	0	0
d_3	1	0	0	0	1	1	0	0
d_4	1	1	0	0	0	1	0	1

FIGURE 4.3 – Exemple de base avant enrichissement.



	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}	w_{16}	w_{17}	w_{18}	w_{19}	w_{20}
d_1	2	3	2	0	1	1	0	0	2	0	1	2	2	1	0	0	0	0	0	0
d_2	0	4	4	2	0	0	1	0	3	0	0	3	3	0	3	1	0	0	1	0
d_3	5	0	0	1	5	5	2	0	0	0	4	0	0	4	1	1	0	0	1	0
d_4	4	2	1	0	3	4	0	2	1	1	3	1	1	3	0	0	1	1	0	1

FIGURE 4.4 – Exemple de base après enrichissement.

4.3 Introduction de la sémantique dans les forêts aléatoires

Dans le même but d'améliorer la classification des messages courts en profitant des relations sémantiques, nous proposons dans la deuxième partie de notre approche un nouveau type de forêts aléatoires. Contrairement aux extra trees proposés par Geurts dans [32] où le caractère aléatoire est renforcé, notre nouvel algorithme consiste à réduire la quantité de l'aléatoire injectée dans les forêts en faveur d'un choix plus sémantique.
Pour introduire la sémantique dans le comportement des forêts aléatoires, nous avons besoin *a priori* d'une ontologie ou d'un réseau sémantique permettant d'identifier les différents liens entre les caractéristiques (les mots composants les messages). Caragea et al. par exemple dans [19] combinent une ontologie avec différents algorithmes d'apprentissage pour améliorer la prédiction des liens d'amitié dans "liveJournal", un réseau social conçu pour les journaux. Caragea et al. construisent au préalable leur ontologie en se basant sur les centres d'intérêts des utilisateurs. Ils partent de l'idée que deux utilisateurs peuvent être amis s'ils partagent le même centre d'intérêt. Cependant, plusieurs relations d'amitiés potentielles peuvent ne pas être prédites si les centres d'intérêt ne sont pas généralisés. En effet, si nous considérons par exemple un utilisateur A qui aime le tennis et un utilisateur B qui aime le football, en apparence aucun lien d'amitié n'est prédit entre ces deux utilisateurs, puisqu'ils ne partagent pas le même centre d'intérêt, mais si nous tenons compte du centre d'intérêt plus général qui est le sport, les deux utilisateurs peuvent être amis. Dans ce but, Caragea et al. ont construit une ontologie hiérarchique de différents niveaux d'abstraction. La construction est faite sur trois étapes :

— la première étape consiste à chercher les descriptions de chaque centre d'intérêt en soumettant des requêtes dans *WordNet en ligne*, *IMDB* (films) et *AWS* (livres). Si aucune description n'est retournée, alors on cherche les synonymes des mots de la requête dans *WordNet en ligne pour les phrases*. Une fois les descriptions collectées, chacune d'entre elles est transformée en un vecteur de mots qui représente un concept dans l'ontologie.
— La deuxième étape consiste à créer le premier niveau d'abstraction de l'ontologie. L'idée initiale est de définir quatre groupes selon la source utilisée. Cependant, puisque le nombre de descriptions des livres collectées par *AWS* est trop grand, alors ce groupe est décomposé en 16 sous groupes suivant le genre des livres. Ainsi, le premier niveau d'abstraction dans l'ontologie est constitué de 19 groupes.
— La dernière étape consiste à construire les autres niveaux d'abstraction de l'ontologie en appliquant un algorithme d'agglomération hiérarchique sur chacun des 19 groupes pour obtenir à la fin 43 niveaux d'abstraction.

Combinée avec différents algorithmes d'apprentissage, cette ontologie a amélioré la prédiction des liens d'amitié dans "Livejournal". Les meilleurs résultats sont obtenus avec les forêts aléatoires. Cependant l'ontologie dans ce travail n'a pas été

intégrée dans les forêts aléatoires, mais elle a été utilisée dans la phase de prétraitement pour construire la matrice représentative des données. En effet, les poids des caractéristiques changent selon l'utilisation de l'ontologie ou pas.
Comme Caragea dans [19], nous sommes partis de l'idée d'avoir un réseau sémantique mais que nous utiliserons dans la construction de la forêt aléatoire et non pas dans la phase de prétraitement. Ce réseau sémantique doit être spécifique à nos caractéristiques. Pour cela, nous avons utilisé LDA [13] une deuxième fois pour construire ce réseau. En effet, une fois les messages enrichis, nous construisons l'espace des caractéristiques puis nous appliquons LDA sur cet espace afin de créer des liens sémantiques entre les caractéristiques. Ainsi, les thèmes obtenus par LDA peuvent être représentés sous forme d'un réseau sémantique (figure 4.5) où nous disposons de deux types de nœuds : les nœuds thèmes représentant un thème et les nœuds mots représentant une caractéristique. Le graphe contient aussi des arcs où chaque arc représente l'appartenance d'un nœud mot à un nœud thème. Ces arcs sont pondérés par l'importance d'une caractéristique dans le thème. Une caractéristique peut appartenir à plusieurs thèmes mais avec des poids différents, d'autres peuvent n'appartenir à aucun thème.

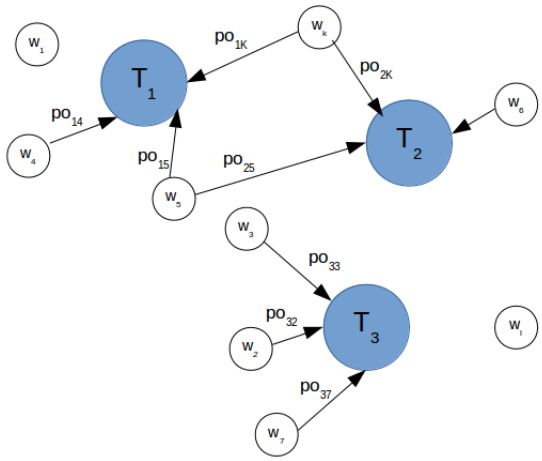


FIGURE 4.5 – Réseau sémantique construit par LDA.

Rappelons le principe de construction d'une forêt aléatoire selon Breiman dans [15]. Pour construire chaque arbre de la forêt deux mécanismes sont utilisés :

— le bagging qui consiste à choisir aléatoirement et avec remise un ensemble de données. L'arbre va être construit ensuite en tenant compte seulement de cet ensemble et
— le Random Feature Selection qui va être utilisé au moment de la construction de chaque nœud de l'arbre, il s'agit de choisir aléatoirement et sans remise un sous ensemble de caractéristiques et de construire le nœud courant en utilisant seulement ce sous ensemble.

Dans notre nouvelle implémentation résumée dans l'algorithme 2, afin de construire des arbres sémantiques, nous ajoutons une nouvelle couche à l'implémentation de base des forêts aléatoires. Chaque arbre n'est plus construit à partir de l'ensemble total des caractéristiques mais plutôt à partir d'un sous ensemble dont les caractéristiques sont sémantiquement proches. Nous appelons ce mécanisme "Semantic Feature Selection" (SFS), il est résumé dans l'algorithme 3. Cette méthode consiste donc à choisir aléatoirement un sous ensemble de caractéristiques de l'ensemble initial de taille L, puis d'utiliser le réseau des thèmes pour l'élargir. Pour chaque caractéristique du sous ensemble choisi, on cherche le thème le plus proche en utilisant la formule 4.2 de calcul de similarité, puis on ajoute tous les mots de ce thème dont le poids dépasse un certain seuil fixé par l'utilisateur. Une fois l'ensemble des caractéristiques obtenu, nous appliquons ensuite le processus du "Random Feature Selection" pour construire les nœuds.
Nous avons tout de même gardé de l'aléatoire dans le choix des caractéristiques du sous ensemble de départ pour deux raisons :

— Conserver la propriété de diversité des forêts aléatoires évoquée par Breiman dans [15].
— Prendre en compte les caractéristiques qui ne sont connectées à aucun thème pour ne pas perdre de l'information.

Algorithme 2 : Arbres sémantiques

Entrée: Car[N] : Ensemble de caractéristiques construites à partir des messages enrichis,
N : nombre total de caractéristiques,
L : nombre de caractéristiques choisies aléatoirement,
Seuil : poids minimum des mots considérés,
Sortie: Forêt sémantique

```
\* Construction du réseau sémantique *\
réseau = appliquerLDASur(car[N])
POUR chaque arbre :
    espaceInitial = choisir aléatoirement L caractéristiques < N
    /* l'élargissement de l'espace de caractéristiques */
    espaceCaracteristiquesFinal ← [ ]
    POUR chaque caractéristique dans espaceInitial :
    /* chercher le thème le plus proche */
    ListeCaracteristiquesAAjouter = SFS(caractéristique,réseau, Seuil)
    SI ListeCaracteristiquesAAjouter <> [ ]
        espaceCaracteristiquesFinal += ListeCaracteristiquesAAjouter
    Construire l'arbre en tenant compte seulement d'espaceCaracteristiquesFinal
```

Le choix initial d'un sous ensemble de caractéristiques puis son enrichissement permettent de garder la diversité au sein des forêts sémantiques. Cependant, si les caractéristiques choisies aléatoirement appartiennent au même thème, les nœuds

Algorithme 3 :Semantic Feature Selection (SFS)

Entrée: caractéristique, réseau, seuil
Sortie: ListeDeCaractéristiques[ ]
  thème = chercher le thème le plus proche (caractéristique, réseau)
  Si thème <>[ ]
    POUR chaque mot dans thème
      Si Poids(mot) > seuil
        ListeDeCaractéristiques + = mot
    retourner (ListeDeCaractéristiques)
  SINON
    retourner [ ]

de l'arbre construit à partir de cet ensemble seront corrélés. C'est pourquoi nous proposons une deuxième version de forêts sémantiques où l'introduction de la sémantique n'est plus faite au niveau des arbres mais plutôt au niveau des nœuds. Pour construire chaque arbre, comme dans les forêts traditionnelles, nous considérons l'espace total des caractéristiques mais nous remplaçons la méthode "Random Feature Selection" par notre nouvelle méthode "Semantic Feature Selection". Le choix de la meilleure caractéristique n'est plus fait sur un ensemble aléatoirement construit mais plutôt sur un ensemble sémantique. Les nœuds d'un arbre ne peuvent jamais être corrélés dans le cas où nous avons un thème dominant puisque cela sera restreint à un seul nœud et non à la totalité de l'arbre. La figure 4.6 illustre les forêts sémantiques niveau nœud.

4.4 Expérimentations et résultats

4.4.1 Benchmark

Afin de valider les forêts sémantiques, Nous les avons testé sur le benchmark "search snippets" collecté par Phan dans [59]. Ce benchmark est composé de deux parties :

— **corpus de messages courts** : construit à partir des 20 à 30 premières réponses données par le moteur de recherche "Google" à des différentes requêtes. Chaque réponse est composée d'une URL, d'un titre et d'une courte description. Le texte court est l'ensemble des mots composant ces trois parties après la phase de nettoyage. Chaque texte est affecté à une classe selon la requête de départ qui l'a engendré. L'ensemble total du corpus contient 8 classes et il est composé de deux parties : apprentissage et test. Les classes, la répartition des données ainsi que quelques statistiques faites sur l'ensemble des textes sont présentées dans le tableau 4.1.

— **base de données universelle** : cette base est composée d'un ensemble d'articles de Wikipédia collectés à partir des réponses à des requêtes contenant des mots clés. L'application de LDA sur cet ensemble a généré 200 thèmes

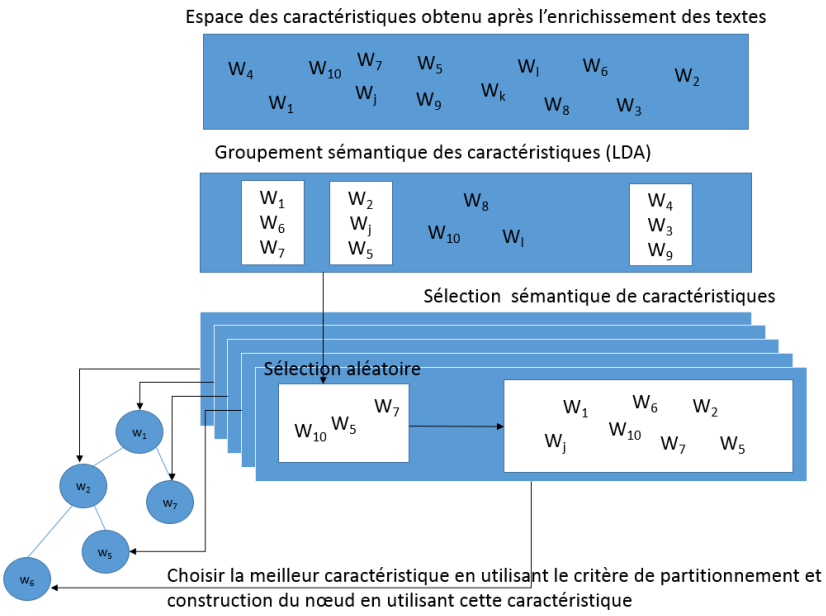


FIGURE 4.6 – Forêts sémantiques au niveau nœud

de 200 mots chacun. Ces thèmes représentent notre base d'enrichissement.

4.4.2 Résultats et interprétations

Dans cette section nous évaluons la valeur ajoutée de nos forêts sémantiques par rapport aux algorithmes traditionnels. Nous avons utilisé comme métrique d'évaluation l'accuracy qui à été définie dans le chapitre 2. C'est le ratio du nombre de messages bien classés par le nombre total de messages de l'ensemble des messages de tests. Nous avons d'abord évalué l'apport de la méthode d'enrichissement en comparant les résultats des forêts aléatoires traditionnelles avant et après enrichissement des données. L'évaluation de cette méthode elle même à été faite sur deux niveaux : le niveau message et le niveau mot. Puis dans une deuxième étape nous avons évalué l'apport de notre algorithme d'apprentissage dans ses deux versions.

4.4.2.1 Évaluation de la méthode d'enrichissement niveau message

Cette expérimentation a un double objectif. Le premier consiste à identifier l'apport de la méthode d'enrichissement au niveau message. Le deuxième consiste à trouver empiriquement le bon nombre de thèmes à ajouter à chaque message (le paramètre p dans la figure 4.2). Nous avons donc fait varier le nombre d'arbres par forêt de 10 jusqu'à 100 par pas de 10. Le critère utilisé pour le partitionnement des nœuds au sein des arbres est "l'indice de Gini". Aucune restriction n'a été faite sur la taille de l'arbre. Il n'y a pas de profondeur maximale exigée, le nombre minimum d'exemples dans les nœuds est égal à un et le nombre minimum pour leur décomposition est égal à deux. Le nombre de caractéristiques K utilisées par le pro-

TABLE 4.1 – Résumé de la distribution des messages courts du corpus "search snippets".

Classes	Apprentissage	Test	% Apprentissage	% Test
Business	1200	300	12%	13%
Computer	1200	300	12%	13%
Culture Art Entertaiment	1880	330	19%	15%
Education science	2360	300	23%	13%
Engineering	220	150	2%	7%
Health	880	300	9%	13%
Politics Society	1200	300	12%	13%
Sport	1120	300	11%	13%
Total	10060	2280	100%	100%

cessus "Random Feature Selection" est égal à la racine carré du nombre total des caractéristiques.

Nous avons lancé la même expérimentation plusieurs fois en variant le nombre de thèmes p ajoutés au message. Il était donc varié de 1 jusqu'à 5. Le tableau 4.2 et la figure 4.7 résument les résultats obtenus. Ces résultats montrent que l'accuracy

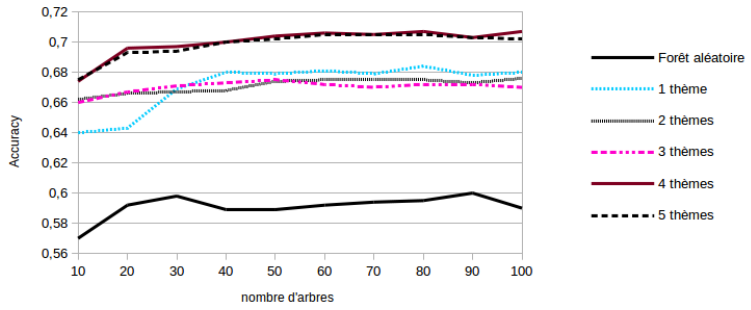


FIGURE 4.7 – Variation de l'accuracy de la classification des messages courts en variant le nombre de thèmes ajoutés par message.

augmente avec le nombre de thèmes jusqu'à 4 thèmes par message. L'accuracy diminue à partir de 5 thèmes. Nous concluons empiriquement que le meilleur nombre de thèmes à ajouter à chaque message est 4. Avec moins de 4 thèmes, l'enrichissement est incomplet, nous ne profitons pas de la totalité de la sémantique qui peut être apportée par la source externe. Au delà de 4 thèmes, nous risquons d'introduire du bruit parce que la distance entre les thèmes et les messages commence à être

TABLE 4.2 – Accuracy de la classification en fonction du nombre d'arbres par forêt et du nombre de thèmes par message

	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
sans enrichissement	0.57	0.592	0.598	0.589	0.589	0.592	0.594	0.595	0.6	0.59
1 thème	0.64	0.643	0.669	0.68	0.679	0.681	0.679	0.684	0.678	0.68
2 thèmes	0.662	0.666	0.667	0.668	0.674	0.675	0.675	0.675	0.673	0.676
3 thèmes	0.66	0.667	0.671	0.673	0.675	0.672	0.67	0.672	0.672	0.67
4 thèmes	0.674	0.696	0.697	0.7	0.704	0.706	0.705	0.707	0.703	0.707
5 thèmes	0.675	0.693	0.694	0.7	0.702	0.705	0.705	0.705	0.703	0.702

importante.
Pour résumer, le niveau message de la méthode d'enrichissement permet d'améliorer nettement la classification des messages du corpus "search snippets" quelque soit le nombre d'arbres par forêt. Les meilleurs résultats sont obtenus quand le nombre de thèmes à ajouter est égal à 4. L'accuracy est passée de 59% à 73% dans le cas de 100 arbres par forêt par exemple.

4.4.2.2 Évaluation de la méthode d'enrichissement sur les deux niveaux

Dans le but d'évaluer la contribution du deuxième niveau d'enrichissement, nous avons lancé de nouveaux tests sur les forêts aléatoires sans enrichissement et avec les deux niveaux d'enrichissement. Les résultats obtenus sont résumés dans le tableau 4.3. Ils montrent une amélioration additionnelle de l'accuracy quand nous avons introduit l'enrichissement au niveau mots et ce quelque soit le nombre d'arbres par forêts. Pour 100 arbres par forêt par exemple, l'accuracy est passé de 70.7% à 76.6%.

TABLE 4.3 – Accuracy de la classification obtenu après les deux niveaux d'enrichissement.

	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
sans enrichissement	0.57	0.592	0.598	0.589	0.589	0.592	0.594	0.595	0.6	0.59
niveau message	0.674	0.696	0.697	0.7	0.704	0.706	0.705	0.707	0.703	0.707
deux niveaux	0.728	0.745	0.753	0.755	0.758	0.758	0.76	0.761	0.76	0.766

4.4.2.3 Évaluation de la méthode d'enrichissement sur d'autres classifieurs

Pour confirmer les résultats obtenus avec les forêts aléatoires, nous avons testé la méthode d'enrichissement sur d'autres algorithmes d'apprentissage tels que les SVM [25], Naive Bayes [69] et Entropie Maximum [7]. Pour les SVM nous avons utilisé le noyau Gaussien et une tolérance à l'erreur $c = 1000000$. Pour Naive bayes nous avons utilisé le modèle Bernoulli. Le tableau 4.4 montre les résultats obtenus avec ces algorithmes d'apprentissage avant et après l'enrichissement.

TABLE 4.4 – Accuracy de classification obtenu avec MaxEnt, SVM et Naive Bayes.

	Entropie Maximum	**SVM**	**Naive Bayes**
sans enrichissement	0.657	0.609	0.493
avec enrichissement	0.734	0.74	0.761

Le tableau montre que l'accuracy augmente quand nous combinons les trois algorithmes avec l'enrichissement des textes. Notre méthode d'enrichissement améliore donc la classification des messages courts quelque soit l'algorithme de classification utilisé.

4.4.2.4 Évaluation des arbres sémantiques et forêts sémantiques niveau nœud

Le but de cette étape est de montrer l'efficacité de nos forêts sémantiques. Nous avons donc lancé une autre série de tests sur le corpus "search snippets". Nous avons remplacé les forêts traditionnelles (RF) par les arbres sémantiques(AS) dans un premier temps, puis par les forêts sémantiques niveau nœud (FS N) dans un deuxième temps. Nous avons fixé $L = 0.9 \times K$ (voir algorithme 2) et le $Seuil = 0$ (voir l'algorithme 3), Enfin, nous utilisons la totalité des mots du thème le plus proche pour assurer un maximum d'élargissement de l'espace de caractéristiques dans le "Semantic Feature Selection". Les résultats obtenus sont résumés dans le tableau 4.5 et la figure 4.8.

TABLE 4.5 – Accuracy pour les forêts traditionnelles, les arbres sémantiques et les forêts sémantiques niveau nœud.

	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
RF	0.57	0.592	0.598	0.589	0.589	0.592	0.594	0.595	0.6	0.59
enrichissement+RF	0.728	0.745	0.753	0.755	0.758	0.758	0.76	0.761	0.76	0.766
Enrichissement+AS	0.73	0.761	0.771	0.776	0.776	0.78	0.784	0.784	0.786	0.789
AS/RF(%)[1]	28.07	28.55	28.93	31.75	31.80	31.76	31.99	31.76	31.00	33.73
AS/Enrichissement(%)	0.27	2.15	2.39	2.78	2.41	2.90	3.16	3.02	3.42	3.00
Enrichissement+FS N	0.766	0.785	0.79	0.787	0.791	0.795	0.794	0.795	0.794	0.795
FS N/RF(%)	34.39	32.60	32.11	33.62	34.30	34.29	33.67	33.61	32.33	34.75
FS N/Enrichissement(%)	5.22	5.37	4.91	4.24	4.35	4.88	4.47	4.47	4.47	3.79
FS N/AS(%)	4.93	3.15	2.46	1.42	1.89	1.92	1.28	1.40	1.02	0.76

TABLE 4.6 – Le nombre de nœuds des arbres des forêts aléatoires, sémantiques et forêts sémantiques niveau nœud.

Arbres	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Moyenne**
Nœuds dans RF	6355	6451	6473	6119	5935	6143	6193	6339	6061	6657	6272
Nœuds dans AS	2957	2883	2657	2653	2879	2851	2873	2717	3037	2753	2826
Nœud dans FS N	1883	1947	1921	1925	1899	1941	1851	1947	1867	1935	1911

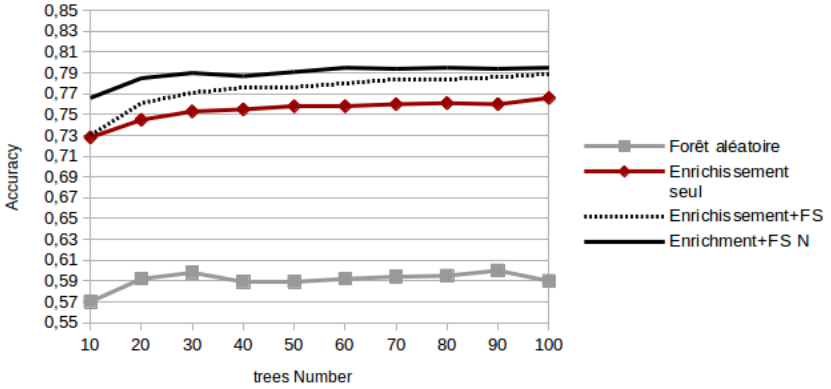


FIGURE 4.8 – Variation de l'accuracy en fonction du nombre d'arbres en utilisant les forêts traditionnelles, les arbres sémantiques et les arbres sémantiques niveau nœuds.

Comme nous pouvons remarquer à partir du tableau 4.5, il y a une amélioration supplémentaire quand les forêts traditionnelles sont remplacées par les arbres sémantiques. Cette amélioration varie entre 2% et 3% en variant le nombre d'arbres par forêt. L'accuracy augmente de 76% jusqu'à 78% pour une forêt de 100 arbres par exemple. Elle augmente de 59% jusqu'au 78% par rapport aux forêts traditionnelles. L'amélioration obtenue dans ce cas varie entre 28% et 33%. Ces résultats sont encore améliorés quand l'introduction de la sémantique se fait au niveaux nœud au lieu du niveau arbre. Pour résumer notre approche sur les forêts sémantiques a nettement amélioré la classification des messages courts par rapport au forêts traditionnelles. L'accuracy a augmenté de 57% jusqu'au 76% pour une forêt de 10 arbres et de 59% jusqu'à 80% environ pour une forêt de 100 arbres, ce qui représente plus que 34% d'amélioration globale. Nos forêts sémantiques ont également battu les scores générés par les autres classifieurs dans le tableau 4.4 même quand ces derniers sont combinés avec l'enrichissement.

4.4.2.5 Taille des arbres

En plus de l'amélioration de l'accuracy, notre méthode présente un deuxième avantage par rapport aux forêts traditionnelles. En effet, nous avons compté le nombre de nœuds des arbres des forêts traditionnelles, sémantiques et des forêts sémantiques niveau nœud pour le cas de 10 arbres par forêt. Le tableau 4.6 résume le nombre de nœuds obtenus dans les 10 arbres pour chacune des trois implémentations. Il montre que la moyenne du nombre des nœuds dans les arbres composant les forêts aléatoires est égale à 6272, alors que celle des arbres sémantiques est à 2826. la taille des arbres sémantiques est réduite à plus de la moitié des forêts aléatoires.

1. SRF/RF(%) le pourcentage d'amélioration de l'accuracy par rapport aux forêts traditionnelles.

Le tableau montre aussi que la moyenne des tailles des arbres des forêts sémantiques niveau nœud est encore plus petite que les arbres sémantiques (1911 nœuds seulement). Les mêmes conclusions sont tirées quand les tests sont lancés sur une forêt de 100 arbres. En effet, la moyenne du nombre de nœuds passe de 6283 pour une forêt aléatoire traditionnelle à 2803 quand il s'agit d'arbres sémantiques et à 1900 pour une forêt sémantique au niveau nœud.
La réduction de la taille des arbres produit une classification des messages plus rapide. Le processus des forêts sémantiques niveau nœud est encore plus rapide puisque la même performance peut être atteinte avec une forêt plus petite comme le montre le tableau 4.5.

4.5 Tests statistiques

Nous avons montré jusqu'à présent que notre méthode est efficace en utilisant comme métrique d'évaluation l'accuracy. Nous confirmons ces résultats en proposant une étude statistique. Le but est de montrer si la différence entre les observations des différents modèles est significative ou elle est simplement due au hasard. Nous avons donc lancé les tests statistiques (ou d'hypothèse) adéquats avec nos données et analysé les résultats obtenus. Dans notre cas, nous cherchons à montrer que la différence entre les résultats obtenus par les forêts traditionnelles et par notre approche de forêts sémantiques est significative. Pour aboutir à un tel résultat, nous suivons la démarche suivante :

- choisir l'hypothèse nulle H_0 et l'hypothèse alternative H_1. L'hypothèse nulle est l'hypothèse de départ, selon laquelle on fixe à priori un paramètre à une valeur particulière, elle représente généralement l'égalité entre deux paramètres statistiques (la moyenne de deux séries de données par exemple). Toute hypothèse différente de H_0 est appelée hypothèse alternative. L'une des deux hypothèses va être acceptée et l'autre va être rejetée ;
- fixer le risque de la première espèce $\alpha \in]0;1[$. C'est un seuil imposé au test de telle façon que la probabilité de rejeter à tort H_0 soit inférieure à ce seuil ;
- déterminer le test élémentaire à utiliser en fonction de la structure des données et de la question posée ;
- calculer la région de rejet en fonction de α et de H_0 ;
- calculer la valeur observée du test (p_value) ;
- conclure : rejet ou acceptation de H_0 au risque α.

Pour montrer qu'une vraie différence existe entre les résultats obtenus par les forêts sémantiques et les forêts aléatoires, nous avons construit plusieurs sous populations de données à partir de la population initiale qui est la base de messages "search snippets". Nous avons généré dans un premier temps 30 paquets de données indépendants. Les messages contenus dans chaque paquet ont été tirés aléatoirement et sans remise. Nous avons construit ensuite deux types de forêts : aléatoires et sémantiques. Nous obtenons ainsi 60 forêts : 30 aléatoires et 30 sémantiques où chaque paquet parmi les 30 est utilisé pour construire une forêt aléatoire et une

autre sémantique. Puis nous avons calculé l'accuracy de ces 60 forêts obtenues en utilisant l'ensemble de test. Le tableau 4.7 représente les deux échantillons obtenus d'accuracy.

TABLE 4.7 – Échantillons d'accuracy obtenu par l'application des RF et des FS sur les 30 sous ensembles de messages.

Echantillon1 : RF	0.299561403509	0.294298245614	0.315350877193	0.277631578947	0.307894736842	0.358333333333
	0.310964912281	0.317543859649	0.338596491228	0.311842105263	0.296052631579	0.239035087719
	0.305701754386	0.277192982456	0.274561403509	0.319298245614	0.290789473684	0.315350877193
	0.29649122807	0.266228070175	0.281578947368	0.323684210526	0.346052631579	0.247368421053
	0.348684210526	0.317543859649	0.304385964912	0.274561403509	0.302192982456	0.286403508772
Echantillon2 : FS	0.638157894737	0.620614035088	0.614035087719	0.598245614035	0.638157894737	0.668421052632
	0.623245614035	0.620175438596	0.6	0.658333333333	0.629385964912	0.652192982456
	0.651754385965	0.649122807018	0.648245614035	0.632894736842	0.593421052632	0.584649122807
	0.672807017544	0.657894736842	0.65350877193	0.644298245614	0.646052631579	0.629385964912
	0.642543859649	0.675438596491	0.65701754386	0.652631578947	0.629385964912	0.638596491228

TABLE 4.8 – Échantillons d'accuracy obtenu par l'application des RF et des FS sur les 10 sous ensembles de messages.

Echantillon1 : RF	0.417543859649	0.393421052632	0.421052631579	0.408771929825	0.45
	0.403947368421	0.396052631579	0.425438596491	0.439473684211	0.425438596491
Echantillon2 : FS	0.706578947368	0.675877192982	0.660964912281	0.68201754386	0.666666666667
	0.688596491228	0.677192982456	0.686403508772	0.676754385965	0.659649122807

Soient μ_1 la moyenne de l'échantillon 1 et μ_2 la moyenne de l'échantillon 2 du tableau 4.7.

- H_0 : $\mu_1 = \mu_2$, les moyennes des deux échantillons ne sont pas différents.
- H_1 :
  - $\mu_1 < \mu_2$
  - $\mu_1 > \mu_2$
  - $\mu_1 \neq \mu_2$
- $\alpha = 0.05$: 5% de probabilité de se tromper.

En se référant à la loi des grands nombres où on peut affirmer que la distribution d'une série de données est normale quand le nombre d'échantillons de ces derniers est supérieur ou égal à 30 et puisque le nombre de nos échantillons (n) est égal à 30, la moyenne des accuracy suit approximativement la loi normale. Comme la distribution de nos données est connu maintenant, nous appliquons ainsi un test paramétrique. Nous disposons de deux échantillons indépendants, le test adéquat à utiliser est le **t_Student** [83]. Il s'agit de calculer la valeur observée de t. t est une variable aléatoire qui suit la loi de Student au degré $n_1 + n_2 - 1$, n_1 et n_2 sont les tailles des deux échantillons, puisque elles sont égales dans notre cas $n_1 = n_2 = n = 30$, la valeur est :

$$t_{obs} = \frac{\mu_1 - \mu_2}{\sqrt{\frac{2S_p^2}{n}}} \tag{4.5}$$

$Sp^2 = \frac{SCE_1 + SCE_2}{n_1 + n_2}$ est la variance commune et SCE est la somme des carrés des écarts.

Pour calculer la p_value en appliquant t_Student sur nos échantillons, nous avons utilisé le logiciel R [1].

La p_value obtenue est largement inférieure à α. L'hypothèse H_0 est donc rejetée, il y a donc une différence significative entre les deux moyennes. L'amélioration que nous apportons n'est pas due au hasard mais plutôt démontrée.

Nous avons également lancé une deuxième expérimentation dans laquelle nous avons construit 10 sous paquets de messages. Le processus de génération des échantillons et le même que le précédent. Les résultats obtenus sont résumés dans le tableau 4.8. L'hypothèse nulle, alternative ainsi que le risque de première espèce ne changent pas. Ce qui change dans cette deuxième expérimentation est le test statistique à utiliser. La loi des grand nombres n'est plus applicable dans ce cas, l'effectif des échantillons étant inférieur à 30. Nous ne connaissons pas à priori la distribution des données, les tests non paramétriques sont applicables dans ce cas et précisément le test de Wilcoxon [31] (deux échantillons aléatoires indépendants). Les deux échantillons obtenus sont des échantillons appariés. En effet, à chaque valeur du premier échantillon lui correspond une valeur du deuxième échantillon. Le test Wilcoxon consiste dans ce cas à calculer la différence entre les valeurs appariées, puis à les classer par ordre croissant des valeurs absolues. Un rang est affecté à chaque différence non nulle puis il s'agit de faire la somme des différences positives w_+ et négatives w_-. $w = min\{w_+, w_-\}$. Le calcul de la p_value dépend de ce w.

1. https ://www.r-project.org/

Nous avons lancé le test Wilcoxon sur nos deux échantillons appariées dans R La P_value obtenue est plus petite que α ce qui signifie qu'il y a une différence entre les résultats obtenus, ce qui confirme l'efficacité de notre méthode par rapport aux forêts traditionnelles.

4.6 Synthèse

Nous avons proposé dans ce chapitre les forêts sémantiques, une nouvelle approche pour améliorer la classification des messages courts. La base de cette approche est de tenir compte de la sémantique des messages tout au long du processus de classification.

L'approche est composée de deux parties majeures : une méthode d'enrichissement au niveau de la phase de prétraitement et un nouvel algorithme d'apprentissage qui se base sur les forêts aléatoires de Breiman.
L'idée d'enrichir les messages courts existe déjà dans la littérature, comme vu dans le chapitre 2. Le motif majeur de cette idée est d'améliorer la qualité des messages courts en réduisant la parcimonie et le manque de contexte, deux limites prédominantes qui rendent les algorithmes traditionnels de classification peu efficaces dans ce contexte. Notre méthode d'enrichissement se distingue des méthodes existantes en proposant un enrichissement sur deux niveaux où nous considérons à la fois la sémantique de chaque mot à part, ce qui nous permet de bien couvrir les différents domaines traités et la sémantique globale du message. L'enrichissement à ce niveau considère le texte comme une seule entité, ce qui résout certains problèmes de polysémie et d'ambiguïté sémantique.
Une fois les textes enrichis, les algorithmes traditionnels sont généralement appliqués. Notre approche diffère des méthodes existantes dans la littérature par la proposition d'un nouveau type de forêts aléatoires qui tient compte de la sémantique interne des messages dans son comportement. Le choix des caractéristiques construisant la forêt ne se base pas uniquement sur l'aléatoire mais aussi sur un choix guidé par la sémantique. Deux versions ont été proposées : choix de caractéristiques sémantiques niveau arbres et niveau nœuds.
Pour valider notre méthode, nous l'avons évaluée sur le corpus "search snippets". Plusieurs tests ont été menés pour évaluer la contribution de chaque partie séparément. Les résultats ont montré que notre méthode est meilleure que les algorithmes traditionnels : plus de 34% d'amélioration par rapport aux forêts aléatoires. Ces résultats ont été confirmés par des tests statistiques. Les expérimentations ont montré également que la taille des forêts sémantiques est beaucoup plus petite que la taille des forêts aléatoires assurant donc une classification plus rapide.

Les forêts sémantiques constituent une approche statique, une fois le modèle de classification crée, il ne change plus au cours du temps. Dans le cas de flux continu de données, cela peut s'avérer problématique. Afin de tenir compte de cette nouvelle connaissance arrivant au cours du temps, nous proposons dans le chapitre suivant une nouvelle approche pour la classification dynamique des messages courts. Cette

approche se base sur l'expertise de l'utilisateur pour mettre à jour le modèle de classification suivant la quantité et la qualité des nouveaux messages.

Chapitre 5

IGLM : Méthode d'apprentissage interactif et générique pour la classification des messages courts

Sommaire

5.1 Introduction **87**
5.2 Abstraction **89**
5.3 Condition de mise à jour du modèle **91**
5.4 Adaptation : mise à jour du modèle de classification **93**
5.5 Expérimentations et résultats **94**
5.5.1 Protocole expérimental 94
5.5.2 Résultats et analyses 95
5.6 Synthèse **100**

5.1 Introduction

Nous avons vu jusqu'à maintenant que le problème de la classification des messages courts est généralement traité d'un point de vue statique. En effet, une phase de pré apprentissage est appliquée pour fournir une représentation mieux adaptée à la problématique du sujet. A l'issue de cette phase, un modèle de classification est construit par un algorithme d'apprentissage qui est ensuite utilisé pour classer le flux des messages arrivant au cours du temps. Ce modèle ne représente plus, à partir d'un certain moment, la réalité des données, surtout dans les domaines qui évoluent rapidement, de nouvelles informations peuvent apparaître et il serait judicieux de réapprendre le modèle en tenant compte de cette nouvelle connaissance afin d'améliorer la performance de la classification.

L'apprentissage actif [79] représente une famille particulière des algorithmes d'apprentissage. Il est utilisé d'une façon générale parce que l'annotation des données consistant à attribuer à chaque message une classe est coûteuse. Nous ne disposons en conséquence que d'une petite base d'apprentissage. Le but de ce type d'apprentissage est d'enrichir cette base par de nouveaux exemples tirés d'une large base de données non annotée pour affiner le classifieur de départ. Les exemples ajoutés doivent appartenir à la zone d'incertitude du classifieur pour qu'ils puissent ajouter

de la précision à ce dernier : ce sont des exemples pour lesquels la probabilité que la classe prédite soit correcte est faible. Généralement, pour déterminer ces exemples, des propriétés du classifieur sont utilisées. Sarawagi et Bhamidipaty les résument dans [68]. Pour SVM [25] le calcul de l'incertitude des exemples dépend inversement de la marge entre l'exemple et l'hyperplan séparateur comme dans [71] ou dans [70]. Pour les arbres de décisions [16], l'incertitude est dérivée de la feuille qui s'est trompée dans la prédiction. Dans le cas de Naive Bayes [69], cette valeur est calculée en se basant sur la probabilité postérieure des classes.

Dans un cas particulier de l'apprentissage actif, l'utilisateur fait partie du processus de classification. Il intervient pour annoter les exemples incertains avant de réapprendre le modèle. Il s'agit de l'apprentissage interactif [68] [29].

Un autre type d'apprentissage dynamique existe dans la littérature : l'apprentissage en ligne. En effet, certaines applications ne disposent pas d'un ensemble d'apprentissage de départ car cet ensemble est construit au fur et à mesure de l'arrivée des données dans le temps. Le modèle de classification dans ce cas, doit être construit de façon incrémentale. Comme nous avons démontré dans le chapitre 3, il est tout à fait possible de construire de façon incrémentale des arbres de décisions [28] ainsi que des forêts aléatoires [66], [1]. Plusieurs défis se posent lors de cette construction. En effet, le modèle généré doit être le plus proche possible de celui que nous aurions obtenu si nous avons disposé de l'ensemble total de données dès le départ. De nouveaux mécanismes tels que le bagging en ligne [55] ont été introduits pour simuler l'apprentissage statique. Un autre défi important est que le modèle doit assurer la classification des données avec une haute performance à n'importe quel moment de sa construction dès le début.

Nous proposons dans ce chapitre une nouvelle méthode pour la classification des messages courts. Par rapport aux méthodes existantes dans la littérature, notre méthode ne se limite pas à construire un modèle de classification statique mais elle propose un processus dynamique qui le met à jour avec l'arrivée de nouveaux messages. Elle se base particulièrement sur l'apprentissage interactif car elle profite de l'expertise de l'utilisateur pour améliorer le modèle. Nous nous sommes inspirés également de l'apprentissage en ligne pour définir quelques propriétés de cette méthode. Enfin, comme elle est dédiée à la classification des textes courts, une représentation matricielle mieux adaptée à notre problématique est construite au sein de cette méthode en tenant compte de la sémantique des textes. Nous appelons cette méthode IGLM.

IGLM, pour "Inreractive Generic Learning Method" est une méthode d'apprentissage à la fois générique et interactive dédiée à la classification des messages courts. Elle est constituée de trois parties majeures :

- une méthode d'abstraction qui représente une phase préliminaire pour améliorer la qualité des messages courts ;
- un mécanisme de calcul statistique de la quantité d'information apportée par les nouveaux messages pour améliorer le modèle de classification ;
- un mécanisme de mise à jour du modèle de classification tenant compte des nouvelles informations.

La figure 5.1 représente le mécanisme général d'IGLM. Nous partons d'un ensemble initial d'apprentissage dont les messages sont pré-traités par la méthode d'abstraction. Nous construisons ainsi une première forêt aléatoire. Une fois que la forêt est utilisée pour la classification, un mécanisme d'évaluation est déclenché à chaque message mal classé corrigé par l'utilisateur. Lorsqu'une certaine condition est vérifiée, nous mettons à jour la forêt. Tous ce processus est accompagné d'une procédure d'évaluation de la performance des différentes forêts ainsi de leurs évolutions au cours du temps.

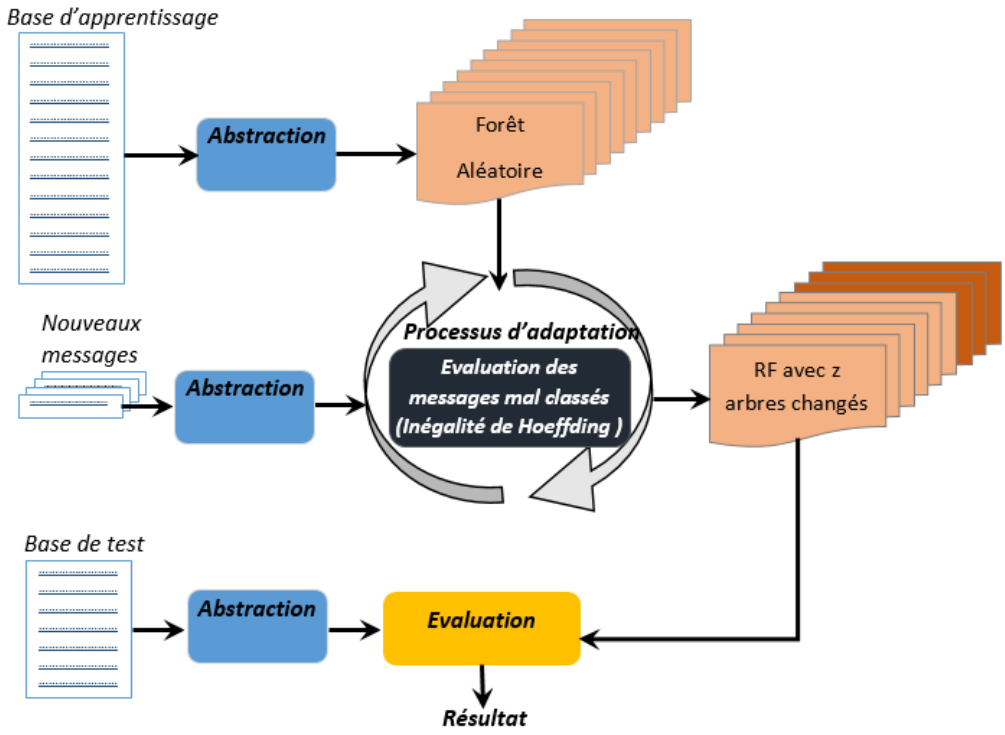


FIGURE 5.1 – Mécanisme général de IGLM.

Dans le reste de ce chapitre, nous allons expliquer en détail les différentes étapes d'IGLM, nous allons ensuite présenter les différentes expérimentations menées pour valider cette approche. Nous finirons ce chapitre par une synthèse.

5.2 Abstraction

L'abstraction est une technique permettant de proposer une représentation des caractéristiques différente de la représentation traditionnelle (Bag of Words). Il s'agit de former des groupes de caractéristiques similaires puis de représenter chaque groupe par sa caractéristique la plus abstraite. Dans le cadre des messages courts l'abstraction est utilisée pour atténuer le problème de parcimonie des données. Elle consiste à regrouper les mots ayant un contexte commun ensemble et à les représenter par un seul mot plus générique.

Mathématiquement, la réduction de la matrice représentative des messages par abstraction peut être formulée comme suit :

— $M_{(m\times N)}$ est la matrice des données avant abstraction ;
— $M'_{(m\times n)}$ est la matrice après abstraction avec $n < N$.

La transformation de M en M' est faite par le biais d'une troisième matrice de correspondance $< caractéristique, abstraction > M"_{(N\times n)}$ où les colonnes sont les mots abstraits et les lignes sont les différentes caractéristiques composant l'espace initial.

$$M'_{(m\times n)} = M_{(m\times N)} \times M''_{(N\times n)}. \tag{5.1}$$

Dans la réalité, il n'est pas possible d'abstraire toutes les caractéristiques. L'espace final va être majoré par ces caractéristiques non abstraites (n") pour éviter toute perte d'information. La matrice finale est donc $M_{(N\times n')}$ où $n' = n + n$".

Nous proposons dans IGLM une nouvelle méthode d'abstraction similaire à celle proposée dans [86], cette méthode se base sur les techniques du "topic models" et en particulier LDA[13]. Pour construire l'ensemble des abstractions, l'idée est exactement la même que celle que nous avons utilisé pour enrichir les messages. Il s'agit de créer une base de connaissance externe spécifique et couvrant les domaines de la base d'apprentissage $D = \{d_1, d_2, .., d_m\}$. Nous appliquons LDA sur cette base pour obtenir un ensemble de thèmes $t = \{t_1, t_2, .., t_n\}$ où chaque thème représente cette fois-ci une abstraction. Le mot caractérisant chaque abstraction est le mot qui a le poids le plus fort. Soit $T = \{T_1, T_2, .., T_n\}$ l'ensemble des abstractions. Chaque message d_i dans D est alors décomposé en un ensemble de mots. pour chaque mot w_{ij} nous cherchons le thème t^* le contenant. Si le mot appartient à plusieurs thèmes, nous choisissons celui dans lequel le mot a le poids le plus fort et w_{ij} est donc remplacé par T^*. Si le mot n'appartient à aucun thème, il reste alors tel qu'il est. La figure 5.2 résume le processus de l'abstraction.

Voici un exemple illustrant le processus de l'abstraction. Soit le message suivant à abstraire :

html wikipedia encyclopedia information file

Pour cet exemple $m = 1$ et $N = 5$, la matrice initiale est $M(1 \times 5)$. Considérons aussi les thèmes suivants :

— Thème 1 : (web, 0.6), (html, 0.3), (wikipedia, 0.1)
— Thème 2 : (data, 0.4), (information, 0.3), (file, 0.3)
— Thème 3 : (book, 0.6), (encyclopedia, 0.4)

n = 3 dans ce cas. Après abstraction le nouveau text est :

web web book data data

Dans cet exemple, l'espace des caractéristiques est réduit de $N = 5$ à $N = 3$ et la nouvelle matrice $M'(1 \times 3)$ est plus petite que l'initiale $M(1 \times 5)$. L'information est dans ce cas plus condensée.

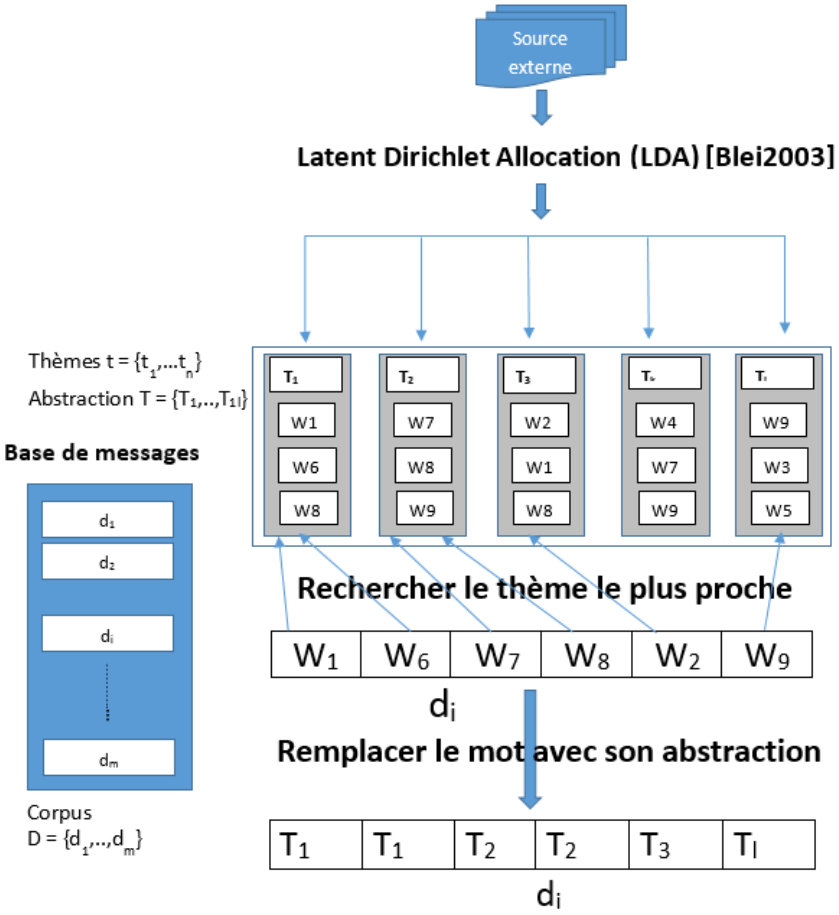


FIGURE 5.2 – Illustration du processus d'abstraction dans IGLM.

Le choix d'utiliser une méthode d'abstraction dans IGLM pour améliorer la représentation matricielle des messages et non pas une méthode d'enrichissement est justifié par la rapidité de l'abstraction par rapport à l'enrichissement. Le but est de ne pas trop alourdir le mécanisme dynamique qui est plus coûteux que le mécanisme statique à cause de l'apprentissage répétitif.

5.3 Condition de mise à jour du modèle

La problématique principale qui se pose pour les forêts aléatoires en ligne est de trouver le bon moment de partitionnement d'un nœud. En d'autres termes, il s'agit de trouver le nombre suffisant d'exemples permettant d'avancer la construction de l'arbre comme en mode statique. Nous rencontrons dans IGLM un défi similaire de mise à jour du modèle même si nous ne nous situons pas dans le même contexte vu que nous disposons d'un ensemble d'apprentissage initial. Une mise à jour du modèle de classification à chaque nouveau message classé peut être coûteuse en terme de

	F_1	F_2	.....	$F_{n'}$
$texte_1$				
$texte_2$				
.....				
$texte_m$				

	F_1	F_2	....	$F_{n'}$
$texte_1$				
$texte_2$				
....				
$texte_m$				
....				
$texte_{m+k}$				

FIGURE 5.3 – Matrice $\mathbf{M}_{(m\times n')}$ et matrice $\mathbf{M}_{(m+k\times n')}$ après mise à jour de la matrice $\mathbf{M}_{(m\times n')}$ avec k textes mal classés.

temps de calcul et pas utile si le message n'est pas très informatif. Trouver le bon moment pour mettre à jour le modèle de classification est une étape cruciale dans IGLM car c'est elle qui va décider de la performance du nouveau modèle par rapport à l'ancien. Nous nous inspirons ici des travaux faits dans l'apprentissage en ligne et spécifiquement de ceux de Domingos dans [28] pour définir la condition de mise à jour du modèle. Il s'agit de trouver le nombre optimal d'exemples à ajouter à la base d'apprentissage avant de mettre à jour le modèle.

Après sa construction, la première forêt est employée pour la classification réelle. A l'arrivé de chaque nouveau message, l'utilisateur intervient pour vérifier et corriger la classification automatique de ce message si elle n'est pas correcte. Nous profitons de cette intervention pour récupérer le message ainsi que sa classe réelle pour améliorer le modèle. Nous considérons que la performance de la forêt courante peut être améliorée en apprenant la modification de la classe associée à ce message. Ce dernier est donc transformé en un vecteur et ajouté ensuite à la matrice courante des données. La figure 5.3 illustre d'une matrice avant ($\mathbf{M}_{(m\times n')}$) et après k messages mal classés ajoutés ($\mathbf{M}_{(m+k\times n')}$).

Après l'ajout de chaque message, la quantité de nouvelle information est évaluée en comparant $\mathbf{M}_{(m\times n')}$ et $\mathbf{M}_{(m+k\times n')}$. Nous utilisons le gain d'information et l'inégalité de Hoeffding [34] [51] pour calculer la différence entre les deux matrices comme dans [28]. Soit $Gain^m(F_i)$ la valeur du gain d'information de la $i^{ème}$ variable F dans la matrice $\mathbf{M}_{(m\times n')}$ et $Gain^{m+k}(F_i)$ le gain d'information de la même variable après K messages mal classés. Les sommes des gains de toutes les variables avant et après l'ajout des données sont respectivement :

$$SomGain(\mathbf{M}_{(m\times n')}) = \sum_{i=1}^{n'} Gain^m(F_i), \tag{5.2}$$

$$SomGain(\mathbf{M}_{(m+k\times n')}) = \sum_{i=1}^{n'} Gain^{m+k}(F_i), \tag{5.3}$$

Soit Δ la valeur absolue de la différence des gains d'information des deux matrices

de données. Elle est définie comme suit :

$$\Delta = |SomGain(\mathbf{M}_{(m+k\times n')}) - SomGain(\mathbf{M}_{(m\times n')})|. \tag{5.4}$$

L'écart entre l'information fournie par la matrice avant et après les k messages est considéré comme suffisant pour mettre à jour le modèle quand Δ dépasse ε. ε est calculée par l'inégalité de Hoeffding. Cette mesure nous permet de répondre à la question suivante "de combien d'itérations avons nous besoin pour déclencher la mise à jour du modèle ?"
Considérons une variable aléatoire r de rang R. Pour mg observations de r avec une moyenne $\bar{r}$, l'inégalité de Hoeffding affirme qu'avec une probabilité $1-\delta$, la moyenne réelle de la variable r est au moins égale à $\bar{r} - \varepsilon$ où :

$$\varepsilon = \sqrt{\frac{R^2 \times \ln \frac{1}{\delta}}{2 \times mg}} \tag{5.5}$$

Une propriété de l'inégalité de Hoeffding démontrée dans [28] est que k nouvelles données sont suffisantes pour mettre à jour le modèle quand la différence entre les gains d'information de la meilleur caractéristique et la deuxième meilleure caractéristique est supérieure à ε. La meilleure caractéristique contient suffisamment d'information pour partitionner le nœud courant. Dans notre cas particulier nous avons besoin d'un aperçu général sur toutes les caractéristiques. Pour cette raison nous utilisons la somme des gains d'information de toutes les caractéristique et nous mettons à jour les forêts quand $\Delta > \varepsilon$. $R = log(c)$, c étant le nombre de classes et mg le nombre total de messages qui s'incrémente avec les nouveaux arrivants.

5.4 Adaptation : mise à jour du modèle de classification

Une fois la condition de mise à jour vérifiée, nous adaptons la forêt actuelle pour qu'elle prenne en considération les nouvelles informations. Le mécanisme d'adaptation consiste à supprimer les arbres les moins performants dans la forêt et à les remplacer par d'autres construits à partir de la nouvelle matrice mise à jour comme illustré dans la figure 5.4.

La première étape de l'algorithme d'adaptation consiste à évaluer la performance de chaque arbre séparément. Un score est attribué pour chacun : il représente la valeur de l'accuracy obtenue en classant l'ensemble de messages de test par cet arbre. Tous les scores sont ensuite triés. Initialement, nous avons opté pour une stratégie où le nombre d'arbres à supprimer est un paramètre fixé par l'utilisateur, ce choix impacte beaucoup plus les forêts peu nombreuses et n'apporte pas une quantité suffisante de modifications pour les forêts de grande taille. C'est la raison pour laquelle nous avons fait le choix que le nombre d'arbres à remplacer z soit proportionnel à la taille de la forêt. Soit :

$$z = 0.1 \times nombre\ total\ des\ arbres \tag{5.6}$$

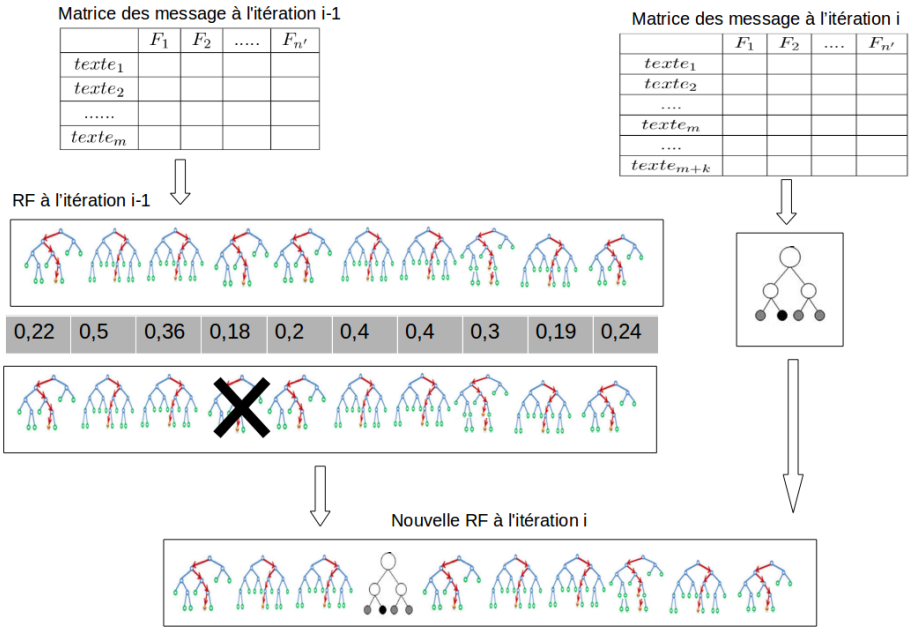


FIGURE 5.4 – Illustration du processus d'adaptation dans IGLM.

Le facteur 0.1 est un compromis entre le nombre d'anciens arbres à garder et le nombre de nouveaux arbres à ajouter. Un grand facteur apporte une grande modification mais nécessite un temps de construction des nouveaux arbres plus important. En contrepartie, un petit facteur engendre une mise à jour peu efficace. Nous avons choisi empiriquement de supprimer 10% des arbres afin d'apporter les modifications nécessaires sans trop dénaturer la forêt initiale.

Une fois que les z arbres à supprimer sont identifiés, nous construisons une nouvelle forêt contenant z nouveaux arbres tout en respectant les principes du Bagging et du Random Feature Selection. Les nouveaux arbres remplacent alors les anciens.

Dans la suite de ce chapitre, nous allons présenter notre protocole expérimental ainsi que les résultats que nous avons obtenus.

5.5 Expérimentations et résultats

5.5.1 Protocole expérimental

Nous avons utilisé le corpus "search snippets" [59] pour évaluer IGLM. Nous avons mis en place un protocole expérimental pour valider les différentes étapes constituant notre approche. L'organisation du corpus est modifiée pour pouvoir tester l'aspect interactif de la méthode. Rappelons que le corpus "search snippets" est composé de deux parties, une base de messages et une base universelle de thèmes qui lui sont sémantiquement proches. L'ensemble des thèmes nous a servi comme ensemble d'abstractions. La partie apprentissage de la base de messages est gardée intacte. La partie test est décomposée en deux sous parties : la première partie est

consacrée à la simulation des flux de messages arrivant au cours du temps pour mettre à jour le modèle. Nous appelons cette partie *ensemble de mise à jour*. La deuxième partie est utilisée pour tester les variétés de modèles générés. Elle est appelée ensemble réel de test.

Les différentes étapes du protocole sont :

— La première itération de IGLM consiste à appliquer l'abstraction sur l'ensemble des messages de la partie apprentissage, puis de construire la première forêt.

— Les textes courts contenus dans *l'ensemble de mise à jour* sont classés un par un par le modèle déjà construit. A chaque fois que le nombre de messages mal classés est atteint, la condition de mise à jour est vérifiée. Un nouveau modèle est alors construit pour remplacer l'ancien et est testé par *l'ensemble réel de test*. Chaque mise à jour est considérée comme une itération.

— la répartition de la partie test originale en un *ensemble de mise à jour* et en un *ensemble de test réel* est répétée aléatoirement 4 fois. Nous obtenons donc 4 benchmarks différents. D_1, D_2, D_3 et D_4.

— Pour chaque expérimentation, IGLM est lancé deux fois pour chacun des 4 benchmarks, une fois avec une forêt de 10 arbres et en remplacent un seul à chaque itération (figure 5.5-a, 5.5-b, 5.5-c, 5.5-d) et une deuxième fois avec une forêts de 100 arbres en remplacent à chaque itération 10 arbres (figure 5.5-a', 5.5-b', 5.5-c', 5.5-d').

— Pour valider les différentes étapes de IGLM, une première série d'expérimentations a été lancée en fixant le nombre de messages mal classés nécessaires pour mettre à jour le modèle à 100 (la condition de mise à jour dans ce cas est un paramètre fixé par l'utilisateur, pas de Hoeffding pour l'instant). Le mécanisme d'adaptation est appliqué dans un premier temps sans tenir compte de l'abstraction (figure 5.5-c, 5.5-c'), puis en ajoutant l'abstraction (figure 5.5-d, 5.5-d'). Nous pouvons ainsi calculer l'apport de chaque étape séparément.

— Une deuxième série d'expérimentations est lancée en introduisant la condition de mise à jour basée sur l'inégalité de Hoeffding dans IGLM pour trouver le bon moment de la mise à jour (figure 5.5-e, 5.5-e').

— La métrique d'évaluation utilisée dans IGLM est encore l'accuracy, qui correspond au ratio des messages bien classés par rapport à la totalité des messages des tests.

5.5.2 Résultats et analyses

Comme expliqué dans le protocole expérimental, la mise à jour du modèle de classification est faite dans un premier temps après chaque 100 messages mal classés. Le but est de mesurer l'amélioration de la classification apportée par le processus d'adaptation par rapport aux forêts traditionnelles et par rapport au mécanisme d'abstraction (première itération). La figure 5.6 représente 4 histogrammes résumant les résultats obtenus sur nos 4 benchmarks D_1, D_2, D_3 et D_4. Chaque histogramme

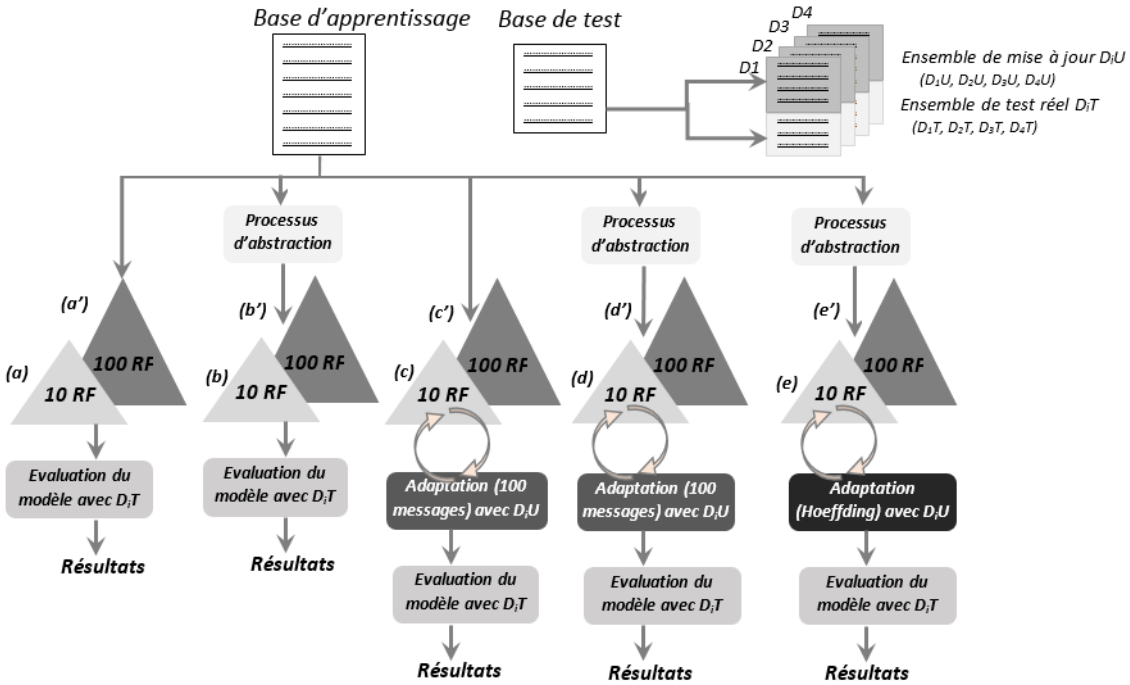


FIGURE 5.5 – Protocole expérimental de IGLM.

contient deux parties, la première représente le résultats d'accuracy pour les différentes combinaisons que nous avons proposées dans le protocole pour 10 arbres, la deuxième représente ceux correspondent aux 100 arbres. Nous pouvons déjà remarquer que sur tous les benchmarks et pour les deux types de forêts, IGLM réalise une importante amélioration dès la première itération. Par exemple, elle dépasse 40% dans le cas du benchmark D_2 avec 100 arbres par exemple. Cette amélioration est réalisée grâce au mécanisme d'abstraction dont le but était de réduire la parcimonie des données pour améliorer la qualité de classification.

L'utilisation de l'adaptation toute seule a montré aussi une amélioration par rapport aux forêts aléatoires traditionnelles. En effet, pour le benchmarck D_3 par exemple ce processus augmente l'accuracy de 34%. Nos meilleurs résultats dans cette première série d'expérimentations sont obtenus en combinant l'abstraction et l'adaptation : nous remarquons une amélioration pour tous les benchmarks (42% pour D_1, 48% pour D_2 et D_3 et 16% pour D_4 pour 100 arbres).

L'évolution de l'accuracy au cours du temps après chaque 100 messages mal classés dans le cas de l'adaptation toute seule ou combinée avec l'abstraction pour les 4 benchmarcks D_1, D_2, D_3, D_4 est résumée dans le tableau 5.1 [1] Nous remarquons que ces résultats sont assez homogènes pour les 4 benchmarcks. Le nombre d'itérations est quasiment le même dans le cas des forêts de même taille. Ces résultats nous

1. Dans le tableau 5.1 : ada fait référence à adaptation et abs fait référence à abstraction.

FIGURE 5.6 – Résumé des résultats obtenus pour les différentes étapes du protocole.

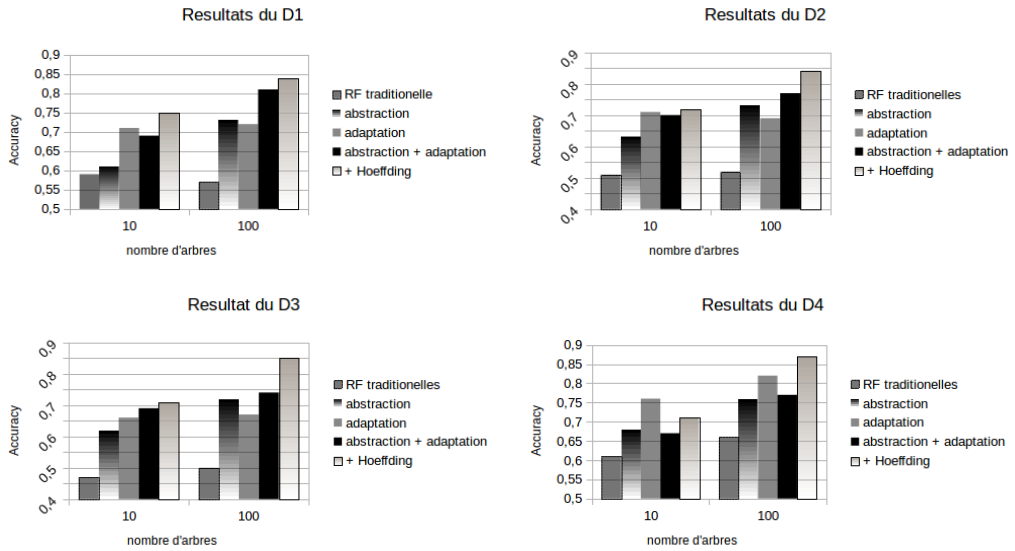


TABLE 5.1 – Evolution de l'accuracy après chaque 100 messages mal classés pour les benchmarks D_1, D_2, D_3, D_4.

iterations	1	2	3	4	5	6	7	8	9
Benchmark D_1									
ada 10 arbres	0,59	0,61	0,62	0,64	0,65	0,66	0,7	0,68	0,71
ada 100 arbres	0,59	0,6	0,6	0,63	0,62	0,64	0,72		
ada + abs 10 arbres	0,61	0,64	0,62	0,65	0,73	0,69	0,67	0,69	
ada +abs 100 arbres	0,73	0,74	0,76	0,77	0,79	0,81			
Benchmark D_2									
ada 10 arbres	0,51	0,5	0,52	0,56	0,56	0,62	0,64	0,65	0,71
ada 100 arbres	0,52	0,52	0,54	0,57	0,58	0,62	0,69	0,69	
ada + abs 10 arbres	0,63	0,66	0,63	0,65	0,63	0,69	0,68	0,7	
ada + abs 100 arbres	0,73	0,73	0,74	0,75	0,77	0,77			
Benchmark D_3									
ada 10 arbres	0,47	0,5	0,49	0,52	0,55	0,59	0,64	0,65	0,66
ada 100 arbres	0,5	0,53	0,56	0,59	0,6	0,64	0,65	0,67	
ada + abs 10 arbres	0,62	0,63	0,66	0,64	0,65	0,66	0,67	0,69	
ada + abs 100 arbres	0,72	0,73	0,72	0,72	0,73	0,74			
Benchmark D_4									
ada 10 arbres	0,61	0,65	0,65	0,68	0,63	0,67	0,7	0,63	0,76
ada 100 arbres	0,66	0,67	0,66	0,68	0,7	0,75	0,8	0,82	
ada + abs 10 arbres	0,68	0,68	0,67	0,67	0,66	0,68	0,71	0,67	
ada + abs 100 arbres	0,76	0,77	0,75	0,77	0,79	0,77			

ont permis de confirmer les mêmes conclusions que nous venons de mentionner ci dessus. Nous remarquons aussi que le nombre d'itérations dans le cas où l'adaptation et l'abstraction sont réunies est plus petit que le nombre d'itérations généré par l'adaptation toute seule. Cette constatation confirme que les deux processus combinés réalisent les meilleures performances. Un nombre d'itérations réduit veut dire que nous avons de moins en moins de messages mal classées, il est plus long de collecter les 100 messages nécessaires pour déclencher la mise à jour, le modèle est donc plus performant. Le tableau montre aussi que les forêts de 100 arbres sont meilleures que celles de 10, en plus de leur meilleure accuracy, leur nombre d'itérations est plus petit.

Regardons maintenant les résultats obtenus en introduisant l'inégalité de Hoeffding. Le nombre de messages après lequel nous lançons l'adaptation n'est plus le même pour toutes les itérations mais il est déterminé grâce à la capacité informative des messages mal classés. Le tableau 5.2 résume les résultats de cette deuxième série d'expérimentations. Il montre pour les 4 benchmarcks et pour 10 et 100 arbres par forêt, l'accuracy des RF traditionnelles, celle de IGLM après la première itération et la dernière itération ainsi que l'accuracy maximale obtenue tout au long du test. La figure 5.7 présente l'évolution de l'accuracy au cours du temps. Le tableau 5.3 est une comparaison entre les résultats obtenus avant et après l'utilisation de l'inégalité de Hoeffding. Ces résultats montrent que l'introduction de l'inégalité de Hoeffding a nettement amélioré la classification. Cette amélioration varie entre 14% et 51% pour les forêts aléatoires de 10 arbres et de 31% jusqu'à 70% pour les forêts aléatoires de 100 arbres par rapport aux forêts traditionnelles quand nous considérons la dernière itération. Elle varie entre 18% et 59% pour les forêts aléatoires de 10 arbres et de 34% jusqu'à 74% pour les forêts aléatoires de 100 arbres par rapport aux forêts traditionnelles quand nous considérons la meilleure itération. Comparée à la mise à jour après 100 messages mal classés, l'inégalité de Hoeffding est plus performante de 14%. Les expérimentations ont montrées également que la moyenne des nombres des textes mal classés nécessaires pour mettre à jour le modèle est de 11 qui est loin des 100 utilisés dans la première série d'expérimentations.

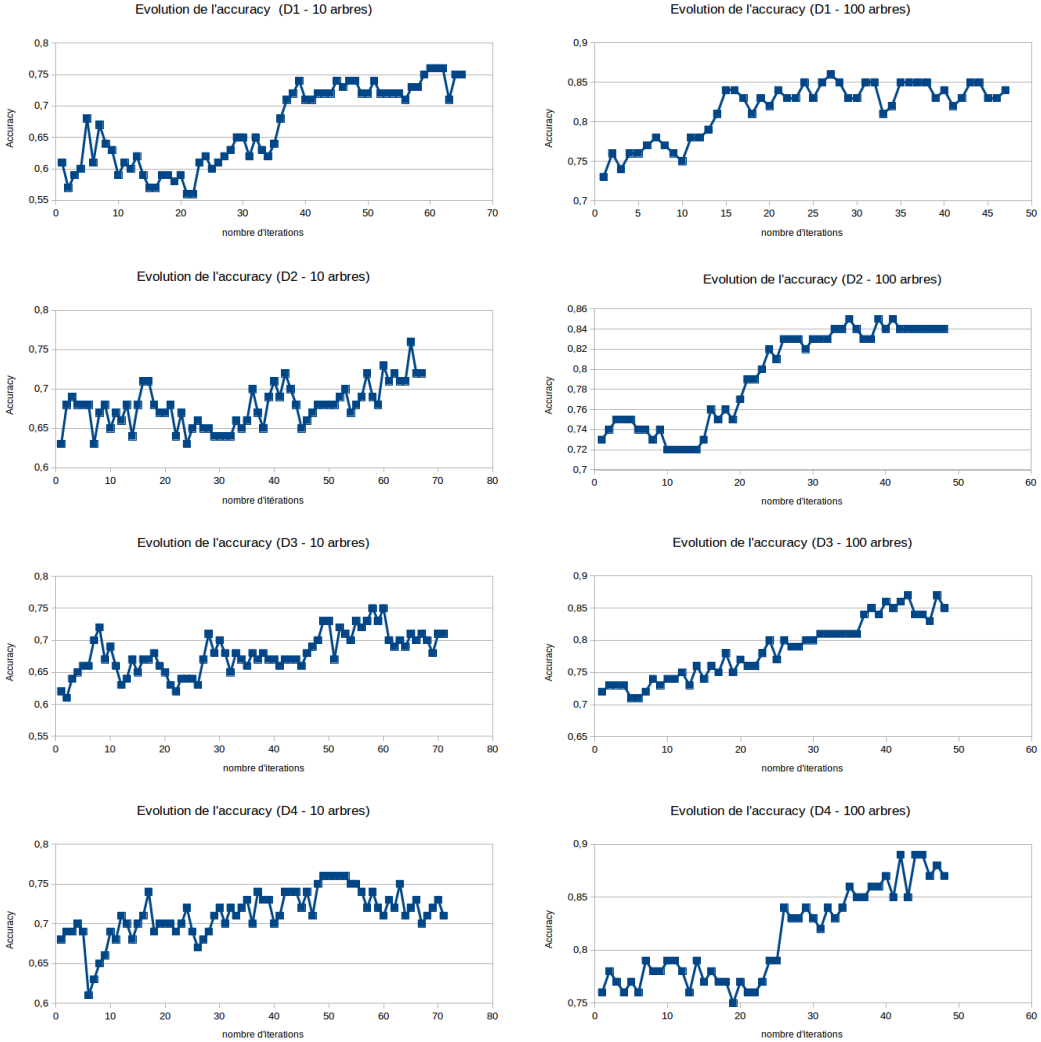
Table 5.2 – Résumé de résultats obtenus en utilisant les forêts aléatoires traditionnelle (T.RF) et IGLM sur les 4 benchmarcks.

	10 arbres				100 arbres			
	T.RF	1ère itér	max	dernière itér	T.RF	1ère itér	max	dernière itér
D_1	0.59	0.61	0.76	0.75	0.57	0.73	0.86	0.84
D_2	0.51	0.63	0.76	0.72	0.52	0.73	0.85	0.84
D_3	0.47	0.62	0.75	0.71	0.5	0.72	0.87	0.85
D_4	0.61	0.68	0.72	0.7	0.66	0.76	0.89	0.87

TABLE 5.3 – Comparaison entre les résultats de IGLM avant et après l'utilisation de l'inégalité de Hoeffding.

Hoeffding		100 messages	
10	100	10	100
0.75	0.84	0.69	0.81
0.72	0.84	0.7	0.77
0.71	0.85	0.69	0.74
0.71	0.87	0.67	0.77

FIGURE 5.7 – Evolution de l'accuracy dans IGLM pour les 4 benchmarks.



Nous avons également étudié l'évolution du paramètre ε (formule 5.5) au cours

du temps pour nos 8 tests. La figure 5.8 montre pour chaque itération i, la valeur de Δ dépassant ε et déclenchant la mise à jour. Nous rappelons que δ est l'écart entre les gains d'information des deux matrices à l'itération $i-1$ et i (formule 5.4). Comme nous pouvons voir à partir de cette figure, ε diminue lentement pour chaque nouveau message entrant. Si nous reprenons la formule 5.5, le rang R et δ sont des valeurs fixes, mg représente le nombre de messages et augmente au cours du temps ce qui explique la diminution d'*epsilon*. Cette diminution permet plus de flexibilité vis-à-vis de la condition de déclenchement de la mise à jour. Cependant cette flexibilité est relative au calcul de l'écart des gains d'informations Δ qui est lui même influencé par la diminution du nombres de messages mal classées vu que le modèle est de plus en plus performant.

5.6 Synthèse

Dans ce chapitre, nous avons proposé IGLM, une nouvelle méthode combinée pour améliorer la performance de la classification des messages courts. Cette approche réunie l'efficacité des méthodes de représentation des données textuelles de courte taille et la puissance des modèles d'apprentissage dynamique, en particulier les forêts aléatoires. IGLM est composé de trois grandes parties :

— une méthode d'abstraction des caractéristiques où une source externe sémantiquement proche de la base d'apprentissage est utilisée pour abstraire les messages initiaux. Cette source est composée d'un ensemble de thèmes où chaque thème est abstrait par son mot le plus représentatif. Ces abstractions servent ensuite à réduire l'espace des caractéristiques en remplaçant chaque mot dans le message par une potentielle abstraction si un lien sémantique existe entre le mot et un thème. Le fait de réduire l'espace des caractéristiques par le biais de la sémantique permet de fournir une matrice plus condensée et d'atténuer le problème de la parcimonie.

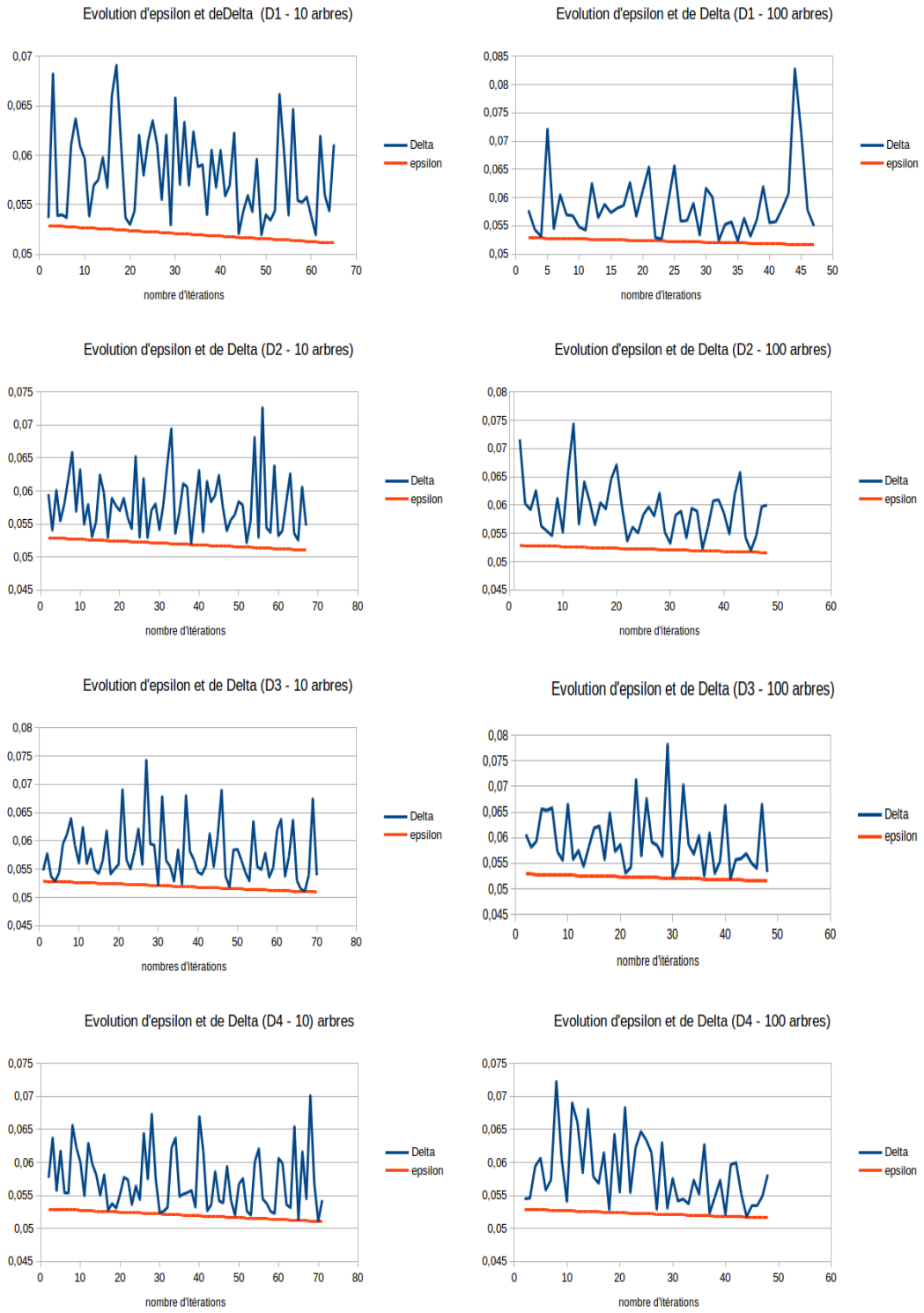
— Une fois qu'une telle représentation est construite, IGLM offre un modèle de classification dynamique où une mise à jour est effectuée en utilisant les nouveaux messages classés et corrigés par l'utilisateur après un ensemble d'itérations. Pour trouver le bon moment de mise à jour du modèle et en se basant sur des prérequis du domaine de l'apprentissage en ligne, nous évaluons la capacité informative des nouveaux messages qui doit dépasser un seuil défini par l'inégalité de Hoeffding.

— Enfin, un mécanisme d'adaptation des forêts aléatoires est mis en place. Il consiste à remplacer les anciens arbres les moins performants par de nouveaux construits en tenant compte de la nouvelle information.

Pour tester notre méthode, nous avons défini un protocole expérimental où le corpus "search snippets" est réorganisé pour simuler l'arrivée des textes au cours du temps. Le corpus est alors réparti en données d'apprentissage, de mise à jour et de test. Nous avons utilisé l'accuracy comme métrique d'évaluation et nous avons tracé son évolution au cours du temps en considérant deux configurations suffisam-

ment distinctes de nombres d'arbres par forêt et en variant la condition de mise à jour du modèle de classification. Les résultats obtenus montrent que notre méthode est efficace, l'accuracy augmente au cours du temps. Ils montrent également que l'utilisation de l'inégalité de Hoeffding pour trouver le bon moment pour effectuer l'adaptation est beaucoup plus judicieuse que d'utiliser un paramètre fixé par l'utilisateur.

FIGURE 5.8 – Évolution des valeurs d'epsilon et de Δ



Chapitre 6

Étude comparative des forêts sémantiques et IGLM

Sommaire

6.1 Introduction **103**
6.2 Points communs et points de différence entre les forêts sémantiques et IGLM **104**
6.2.1 Points communs 104
6.2.2 Points de différence 104
6.3 Accuracy **105**
6.4 Taille des Forêts **108**
6.5 Temps de calcul et complexité **110**
6.5.1 Temps de calcul et complexité de l'enrichissement et de l'abstraction 110
6.5.2 Temps de calcul des algorithmes d'apprentissages 112
6.6 Parcimonie ou "Sparsity" **113**
6.7 Discrimination des caractéristiques **116**
6.8 Synthèse **118**

6.1 Introduction

Nous avons proposé dans les deux chapitres précédents 4 et 5 les forêts sémantiques (FS) et la méthode d'apprentissage interactif générique (IGLM), deux approches différentes pour la classification des messages courts.

Les forêts sémantiques sont composées de deux parties majeures : l'enrichissement de messages courts (Enrich) et la forêt sémantique dans ses deux versions, arbres sémantiques (AS) et forêt sémantique au niveau des nœuds (FS N). Les algorithmes 4, 5 et 6 décrivant cette méthode sont rappelés à la fin de ce chapitre.

IGLM est composée de trois grandes parties : l'abstraction des données (Abs), la condition de mise à jour du modèle et l'adaptation (Adapt). L'algorithme 7, présenté à la fin de ce chapitre, résume le fonctionnement d'IGLM.

Ces deux approches ont été testées et validées séparément. Nous proposons dans ce chapitre de les comparer pour mieux expliquer les différents choix faits. Les deux approches s'intéressent au sujet en question sous deux angles différents. En effet, la première propose une solution statique avec un nouvel algorithme d'apprentissage

basé sur les forêts aléatoires [15], alors que la deuxième traite le sujet d'un point de vue dynamique où le modèle évolue au cours du temps. Cependant, elles restent comparables car elles possèdent un certain nombre de points communs, notamment la façon de se comporter avec la nature des données (courte taille, parcimonie et manque de contexte). Nous identifions donc dans ce chapitre les points communs et les différences entre ces deux méthodes en menant une étude comparative suivant plusieurs axes tels que l'accuracy, la taille des forêts, le temps de calcul et la parcimonie des données. Cette étude se base sur les résultats obtenus sur le corpus "search snippets" [59].

6.2 Points communs et points de différence entre les forêts sémantiques et IGLM

6.2.1 Points communs

Nous avons identifié 5 points communs entre les forêts sémantiques et IGLM. Nous les résumons dans les points suivants :

— les deux méthodes agissent à la fois dans l'étape prétraitement et dans l'étape d'apprentissage du processus d'extraction de la connaissance à partir des données.

— Les deux méthodes proposent des solutions pour améliorer la qualité des messages en diminuant leur parcimonie ;

— le même algorithme d'apprentissage est utilisé dans les deux méthodes. Il s'agit des forêts aléatoires ;

— les deux méthodes nécessitent des documents externes pour l'enrichissement ou pour l'abstraction ;

— les deux méthodes sont sensibles à la qualité des documents collectés. En effet, ces documents doivent être spécifiques aux messages courts traités et couvrir le plus possible leurs différents domaines pour assurer un enrichissement ou une abstraction efficaces.

6.2.2 Points de différence

Trois points majeurs de différence entre les forêts sémantiques et IGLM sont identifiés. Ils sont détaillés dans les points suivants :

— la forêt sémantique est une approche statique puisqu'une fois l'enrichissement fait et la forêt sémantiquement créée, le modèle ne change pas au cours du temps. IGLM est au contraire une approche dynamique puisque les forêts changent au cours du temps avec l'arrivée de nouvelles données et elles s'adaptent aux corrections faites par l'utilisateur ;

— l'amélioration de la qualité des messages dans les deux approches se base sur deux idées opposées : l'une se base sur l'augmentation de la taille du vecteur représentant un message et l'élargissement de l'espace des caractéristiques afin d'ajouter de l'information dans le vecteur (enrichissement). L'autre se

base sur une réduction de la dimension des caractéristiques et la taille du vecteur représentant un message afin de condenser la matrice de données (abstraction) ;

— en se basant sur la matrice représentative des messages courts où les colonnes sont les caractéristiques et les lignes sont les messages, on peut considérer l'approche sémantique comme une vision horizontale puisqu'elle travaille sur **la nature des caractéristiques** (les liens sémantiques entre elles). En suivant ce raisonnement, nous pouvons voir IGLM comme une approche verticale puisqu'on manipule **la quantité d'information apportée à chaque caractéristique** avec l'arrivée de nouvelles données.

6.3 Accuracy

Comme nous venons de dire dans l'introduction, même si le même benchmark "search snippets" a été utilisé pour valider les deux approches, les protocoles expérimentaux implémentés ne sont pas les mêmes. Une comparaison directe entre les approches n'est donc pas possible. En effet, pour tester IGLM la partie test du corpus a été décomposée en deux sous parties *ensemble de mise à jour* et *ensemble de test réel*. L'*ensemble de mise à jour* sert à simuler l'intervention de l'utilisateur pour corriger les erreurs de classifications. Néanmoins, dans les deux approches, il y a deux parties pour lesquelles l'accuracy est comparable. Il s'agit de l'enrichissement et de l'abstraction : deux visions complètement opposées pour résoudre la même problématique "la parcimonie des messages courts". Les résultats présentés dans le chapitre 5 concernant l'abstraction sont obtenus en utilisant seulement *l'ensemble de test réel* et non pas l'ensemble total de test. Nous calculons dans cette section l'accuracy obtenue par l'abstraction en utilisant l'ensemble total de test pour qu'elle soit comparable avec celle obtenue avec l'enrichissement. Le tableau 6.1 et la figure 6.1 résument les résultats obtenus sur les forêts aléatoires traditionnelles (en utilisant la représentation "Bag Of Words") et en les combinant avec l'enrichissement et l'abstraction. Le nombre d'arbres par forêt varie entre 10 et 100 arbres par pas de 10.

TABLE 6.1 – Variation de l'accuracy en fonction du nombre d'arbres par forêt pour l'enrichissement et l'abstraction.

	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
Bag Of Words	0.57	0.592	0.598	0.589	0.589	0.592	0.594	0.595	0.6	0.59
Enrichissement	0.728	0.745	0.753	0.755	0.758	0.758	0.76	0.761	0.76	0.766
Abstraction	0.644	0.680	0.690	0.703	0.714	0.715	0.721	0.724	0.723	0.725

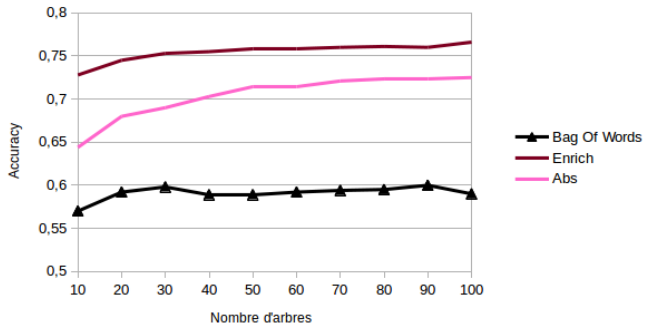


FIGURE 6.1 – Variation de l'accuracy en fonction du nombre d'arbres pour l'enrichissement, l'abstraction et les forêts traditionnelles.

Comme nous pouvons observer dans le tableau 6.1 et la figure 6.1, l'enrichissement et l'abstraction sont tous les deux meilleurs que les forêts traditionnelles. Nous constatons aussi que l'accuracy augmente en augmentant le nombre d'arbres pour chaque méthode. Les résultats montrent aussi qu'en terme d'accuracy l'enrichissement est plus performant que l'abstraction mais avec une légère différence de 5% pour 100 arbres. Une explication possible pour justifier cette différence est que les données obtenues après enrichissement sont moins creuses ("sparse") que celles après abstraction. La vérification de cette hypothèse sera étudiée dans la section 6.6.

Afin de mieux détailler les résultats obtenus dans le tableau 6.1, nous avons réalisé quelques statistiques à partir de ces résultats. Rappelons que le benchmarck "search snippets" est composé de 8 catégories. Nous avons calculé pour chaque catégorie le nombre de messages mal classés dans plusieurs cas. Le tableau 6.2 représente le nombre de messages mal classés dans le cas de l'enrichissement (Enrich) et de l'abstraction (Abs). Il représente aussi le nombre de messages mal classés par l'enrichissement et bien classés par l'abstraction (Abs > Enrich) et finalement, les messages mal classés par les deux en même temps (les deux). Nous traçons l'histogramme de l'erreur de classification par catégorie. Rappelons que l'erreur de classification est définie comme étant le pourcentage du nombre de messages mal classés par rapport au nombre total de messages de test. Cette mesure est l'inverse de l'accuracy.

A partir de la figure 6.2 nous remarquons que même si l'abstraction et l'enrichissement sont deux idées basées sur deux concepts contradictoires, leurs résultats obtenus sont homogènes. En effet, les deux méthodes se trompent le plus et le moins sur les mêmes classes ("Engineering" et "Culture Art Entertaiment" respectivement). La figure révèle aussi un grand écart d'erreur de classification de ces deux classes. Ceci s'explique par le fait que les messages dans le corpus "search snippets" ne sont pas répartis sur les différentes classes d'une façon équiprobable, la classe "Engineering" est très sous représentée par rapport à la classe "Culture Art Entertaiment". Une potentielle amélioration peut être envisagée en utilisant les forêts

TABLE 6.2 – Le nombre de messages mal classés par catégorie du corpus "search snippets"

Catégorie	Enrich	Abs	Abs>Enrich	les deux	total messages
Business	69	91	29	20	300
Computer	77	118	23	31	300
Culture Art Entertaiment	60	72	28	16	330
Education Science	56	77	15	21	300
Engineering	92	140	4	45	150
Health	87	119	32	25	300
Politcs Society	102	153	23	42	300
Sport	70	91	16	28	300
Total	613	861	170	228	2280

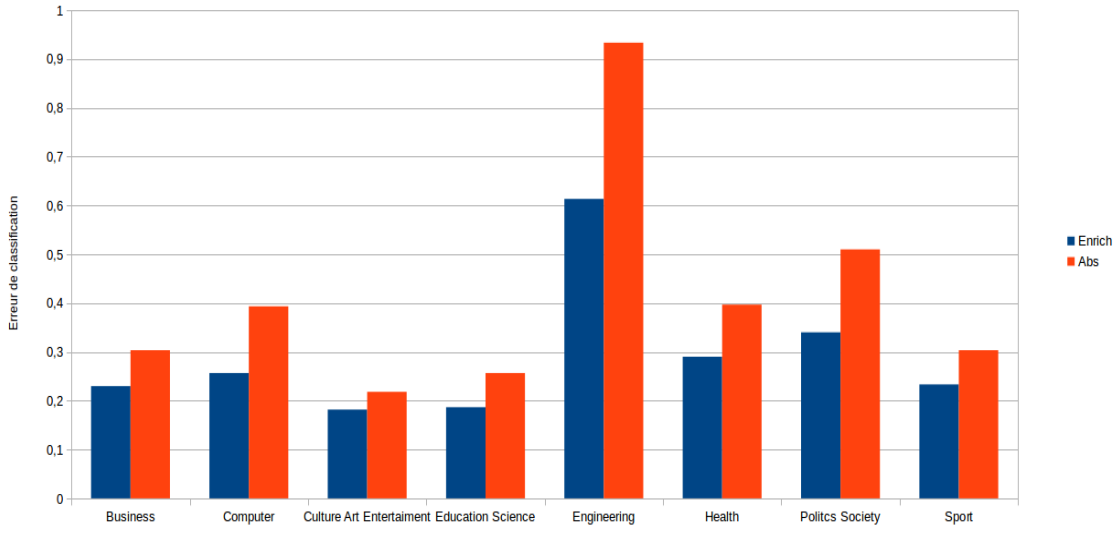


FIGURE 6.2 – Erreur de classification de l'enrichissement et de l'abstraction par catégorie.

pondérées ou en les combinant avec les forêts sémantiques.

Nous remarquons aussi à partir du tableau 6.2 que 7% des messages mal classés par l'enrichissement ont été bien classés par l'abstraction. Même si l'accuracy générale de l'enrichissement est meilleur que celle de l'abstraction, cette dernière arrive à bien classer des messages pour lesquels l'autre méthode à échoué. Une hybridation des deux méthodes en une seule en définissant par exemple une heuristique permettant de choisir la méthode la plus adéquate peut donc augmenter l'accuracy et minimiser l'erreur de classification.

6.4 Taille des Forêts

Pour calculer les tailles des forêts générées par nos algorithmes, nous avons compté le nombre de nœuds de chaque arbre. Le tableau 6.3 résume le nombre de nœuds des arbres de la forêt traditionnelle, après enrichissement et après abstraction. Nous étudions dans cette section les forêts à 10 arbres.

TABLE 6.3 – Nombre de nœuds de chaque arbre pour les trois types de forêts.

Arbre	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**moyenne**
Bag Of Words	6355	6451	6473	6119	5935	6143	6193	6339	6061	6657	6272
Enrichissement	2813	2925	2657	2819	2731	2741	2675	2863	2789	2837	2793
Abstraction	8451	7825	8003	8019	8153	7961	7925	8275	8161	8185	8096

Le tableau 6.3 montre que les 10 arbres construits après enrichissement sont beaucoup moins complexes (plus petits) que les arbres construits par les forêts traditionnelles. En effet, la moyenne des nombres de nœuds des 10 arbres obtenue après enrichissement est inférieure à la moitié de la moyenne des nombres de nœuds des 10 arbres obtenue par les forêts traditionnelles. Pour l'abstraction, les arbres construits sont plus complexes que les arbres obtenus avec les forêts traditionnelles. Une explication possible est que la complexité des arbres dépend inversement de la taille de l'espace des caractéristiques à partir duquel l'arbre est construit. En effet, l'espace fourni après l'enrichissement est beaucoup plus large que celui du "Bag of Word" traditionnel, ce dernier est plus large que celui résultant de l'abstraction. De ce fait, nous pouvons déduire que les arbres les moins complexes sont ceux issus de l'espace le plus large. Ils convergent plus vite car il y a plus de choix de caractéristiques.

Le tableau 6.4 résume aussi le nombre des nœuds des 10 arbres obtenu pour la forêt sémantique et la forêt sémantique au niveau des nœuds.

Le tableau 6.4 montre que par rapport à la forêt enrichie, la forêt sémantique au niveau des nœuds est beaucoup moins complexe. Ceci la rend plus rapide dans la classification et plus efficace.

Nous avons aussi compté le nombre de nœuds des arbres créés dynamiquement dans IGLM. Pour ce dernier cas nous avons fixé 10 arbres par forêt et une mise à jour après chaque 100 messages mal classés. Le tableau 6.5 présente le nombre de

TABLE 6.4 – Nombre de nœuds par arbre des forêts sémantiques et sémantiques au niveau nœud.

Arbre	1	2	3	4	5	6	7	8	9	10	moyenne
RF	6355	6451	6473	6119	5935	6143	6193	6339	6061	6657	6272
AS	2957	2883	2657	2653	2879	2851	2873	2717	3037	2753	2826
FS N	1883	1947	1921	1925	1899	1941	1851	1947	1867	1935	1911

nœuds des arbres de la forêt initiale et des forêts obtenues après les mises à jour. Les cases colorées en jaune contiennent le nombre de nœuds des arbres qui ont été supprimés et les cases colorées en vert contiennent le nombre de nœuds des arbres ajoutés.

TABLE 6.5 – Nombre de nœuds par arbre des forêts d'IGLM .

Arbre	1	2	3	4	5	6	7	8	9	10	moyenne
Forêt initiale	8451	7825	8003	8019	8153	7961	7925	8275	8161	8185	8096
2ème itération	8451	7825	8003	8361	8153	7961	7925	8275	8161	8185	8130
3ème itération	8451	7825	8277	8361	8153	7961	7925	8275	8161	8185	8157
4ème itération	8451	7825	8277	8371	8153	7961	7925	8275	8161	8185	8158
5ème itération	8455	7825	8277	8371	8153	7961	7925	8275	8161	8185	8159
6ème itération	8455	9013	8277	8371	8153	7961	7925	8275	8161	8185	8278
7ème itération	8455	9013	8277	8747	8153	7961	7925	8275	8161	8185	8315
8ème itération	8455	9013	8277	8747	8153	9045	7925	8275	8161	8185	8424

En analysant le tableau 6.5, nous remarquons que la moyenne des nombres de nœuds par arbre de la forêt obtenue dans IGLM est plus grande que celle des arbres de la forêt traditionnelle. En effet, la première itération de l'algorithme consiste à appliquer le principe de l'abstraction sur les données et nous avons déjà démontré que la procédure d'abstraction augmente le nombre de nœuds par arbre. L'adaptation ne permet pas de diminuer le nombre de nœuds des arbres non plus. En étudiant

la variation de cette moyenne d'une itération à une autre, nous remarquons qu'elle augmente au fur et à mesure des mises à jour. En parallèle l'accuracy s'améliore également avec les mises à jour.

Pour conclure, le facteur principal qui permet de déterminer la complexité des arbres dans IGLM ou dans les forêts sémantiques est la méthode utilisée pour représenter les messages courts : l'abstraction pour IGLM et l'enrichissement pour les forêts sémantiques. Ainsi, comme l'enrichissement a permis de diminuer la complexité des arbres et comme l'abstraction l'a augmentée, alors les arbres des forêts sémantiques sont moins complexes que ceux d'IGLM.

6.5 Temps de calcul et complexité

Dans cette section nous nous intéressons aux temps d'exécution de nos programmes. Nous allons les comparer pour déterminer le plus rapide d'entre eux et dans quels cas chacune de ces méthodes est plus efficace. Soit u une unité de temps. Le temps de calcul de chaque programme va être exprimé en fonction de u.

6.5.1 Temps de calcul et complexité de l'enrichissement et de l'abstraction

La première partie consiste à comparer l'enrichissement et l'abstraction. Nous avons donc lancé ces deux procédures sur l'ensemble total des messages du corpus "search snippets" et nous avons calculé le temps pris pour chacune des deux. Le tableau 6.6 résume les résultats obtenus.

TABLE 6.6 – Temps d'exécution de l'enrichissement et l'abstraction

	TE total	TE par message	TE des messages d'apprentissage	TE des messages de test
Enrichissement	2652 u	0.21 u	2162 u	490 u
Abstraction	13 u	10^{-3} u	11 u	2 u

Comme nous pouvons constater, l'abstraction est beaucoup plus rapide que l'enrichissement. Le tableau 6.6 montre que le temps d'exécution de l'enrichissement représente 204 fois le temps d'exécution de l'abstraction sur le même corpus "search snippets".
Même si l'enrichissement est plus efficace dans la classification des messages comme cela est démontré dans la section 6.3, il reste difficile à appliquer dans un contexte temps réel dans lequel les messages arrivent en série et leur temps de classification ne doit pas dépasser un laps de temps très court. Dans ce cas l'abstraction est plus

adaptée, elle est utilisable dans un contexte de temps réel même si elle ne garantie pas forcément la même accuracy que l'enrichissement.

Pour mieux comprendre pourquoi l'abstraction est plus rapide que l'enrichissement, nous étudions la complexité de l'algorithme de chacune des deux méthodes. Soient :

— **p** le nombre de messages du corpus ;
— **w** le nombre de mots du plus grand message dans le corpus ;
— **m** le nombre de thèmes ;
— $\mathbf{w_m}$ le nombre de mots dans un thèmes.

L'abstraction consiste à parcourir tous les mots de tous les messages du corpus et pour chaque mot on cherche le thème le plus proche en parcourant tous les mots de tous les thèmes. La complexité de la recherche est donc :
$\mathbf{O(p.w.m.w_m)+ O(p.w.m)}$.
Une fois que nous avons trouvé le bon thème, il s'agit de changer le mot par son mot abstrait. Cette opération est faite en $\mathbf{O(1)}$.

La complexité de l'abstraction est donc :
$$\mathbf{O(p.w.m.w_m)+ O(p.w.m)}.$$

L'enrichissement se fait sur deux niveaux (mot et message) :
niveau mot : comme pour l'abstraction, il s'agit de parcourir les messages du corpus puis les mots de chaque message, ensuite, pour chaque mot il faut parcourir tous les mots de tous les thèmes afin de calculer le score de chaque thème par rapport à ce mot (un tableau T de taille m où $T[i]$ représente le score du $i_{ème}$ thème). Cette opération est donc faite en $\mathbf{O(p.w.m.w_m)}$. Il s'agit ensuite de trouver le thème le plus proche (minimum de $T[i]$). Cette opération est faisable en $\mathbf{O(p.w.m)}$. Une fois le bon thème trouvé, l'algorithme ajoute tous les mots de ce thème au mot actuel. La complexité de cette opération est $\mathbf{O(p.w.w_m)}$.
La complexité de l'enrichissement au niveau mot est :

$$\mathbf{O(p.w.m.w_m)+O(p.w.m)+ O(p.w.w_m)}\,;$$

niveau message : La recherche des thèmes est un peu différente de celle de l'abstraction et l'enrichissement niveau mot. En effet, il faut trouver les **k** thèmes les plus proches du message. Pour calculer la distance de chaque thème à chaque message, il faut parcourir pour chaque message tous ses mots et tous les mots des thèmes. Le coût de cette recherche est donc $\mathbf{O(p.w.m.w_m)}$. Le coût de l'ajout des thèmes au messages initiaux est donc : $\mathbf{O(p.k.w_m)}$.
La complexité de l'enrichissement au niveau message est :

$$\mathbf{O(p.w.m.w_m)+O(p.k.w_m)}.$$

La complexité de l'enrichissement est donc :
$$\mathbf{2.O(p.w.m.w_m)+O(p.(w.(m+w_m)+k.w_m))}$$

Comme nous pouvons le constater, l'algorithme de l'enrichissement est plus complexe que celui de l'abstraction, puisque dans l'abstraction il n'y a que le coût de recherche des thèmes. Le coût de remplacement du mot par le mot abstrait est négligeable. Tandis que pour l'enrichissement, en plus du doublement du coût de la recherche, il y a aussi le coût d'ajout des thèmes.

6.5.2 Temps de calcul des algorithmes d'apprentissages

Nous calculons maintenant le temps d'exécution des forêts sémantiques et d'IGLM. Le tableau 6.7 résume les résultats obtenus pour 10 arbres par forêt.

TABLE 6.7 – Le temps d'exécution des forêts sémantiques et d'IGLM

		Temps d'exécution
Forêts Sémantiques	AS	2603 u (2162 u pour l'enrichissement)
	FS N	2839 u (2162 u pour l'enrichissement)
IGLM	100 messages mal classés	12 u (11 u pour l'abstraction)
	Hoeffding	1797 u (11 u pour l'abstraction)

6.5.2.1 Forêts sémantiques

Le temps total de construction de la forêt sémantique est composé de :

— **temps de la méthode d'enrichissement** : 2162 u.

— **temps de construction des arbres sémantiques** : 441 u.
Comparés aux arbres traditionnels, les arbres sémantiques présentent un temps additionnel de calcul. Il représente :

1. le temps d'application de LDA sur l'ensemble des caractéristiques pour les regrouper sémantiquement. Cette opération dure plus que 420 u. C'est l'opération la plus coûteuse.
2. le temps de recherche et d'ajout des thèmes les plus proches à chaque caractéristique de l'ensemble de départ. Cette opération est répétée autant de fois que le nombre des arbres. Le temps de calcul de cette opération ne dépasse pas 3 u dans ce cas.

— **temps de construction des nœuds sémantiques** : 667 u.
La construction des nœuds sémantiques suit le même principe que la construction des arbres sémantiques. Il s'agit de choisir aléatoirement un petit ensemble de caractéristiques puis d'élargir cet ensemble avec des thèmes sémantiquement proches. Cette opération ne se fait plus au niveau de chaque arbre mais se répète plutôt pour chaque nœud, ce qui rend la construction des forêts sémantiques au niveaux nœud plus coûteuse. Pour optimiser ce processus, nous avons construit au préalable la matrice de correspondance $M_{(mots,thèmes)}$. Cette matrice indique le thème le plus proche pour chaque

caractéristique de l'ensemble entier. Pour la création de chaque nœud, il suffit de consulter pour chaque mot i dans l'ensemble initial son thème le plus proche $M[i]$ et d'ajouter ses éléments à l'ensemble initial. Pour résumer, la forêt sémantique au niveau nœud présente les temps additionnels suivants par rapport aux forêts traditionnelles :

1. le temps de regrouper sémantiquement les caractéristiques en utilisant LDA
2. le temps de construction de la matrice de correspondance caractéristiques / thèmes.
3. le temps d'ajouter les mots des thèmes pour élargir l'ensemble des caractéristiques candidates pour la construction de chaque nœud.

6.5.2.2 IGLM

Le temps d'exécution d'IGLM est composé de :

— **temps de l'abstraction** : 11 u

— **temps de la mise à jour après 100 messages mal classées** : $1u$. Ce temps comprend les huit mises à jour de la forêt de 10 arbres. Soit $1/8u$ pour chaque mise à jour. C'est le temps de la recherche et la suppression du mauvais arbre et la création et l'ajout du nouvel arbre. La mise à jour dans le cas où la condition dépend d'un paramètre fixe est peu coûteuse en temps.

— **temps de la mise à jour en utilisant l'inégalité de Hoeffding** : 1786 u.
L'utilisation de l'inégalité de Hoeffding rend l'application beaucoup plus lente, car dans cette version le calcul du gain d'information de chaque caractéristique de la représentation matricielle est introduit pour chaque message mal classé. Cette opération prend 3 u par message. Un autre facteur qui entraîne une lenteur de l'application par rapport à la version précédente est le nombre de mises à jour qui dépasse largement 8 fois, puisque la moyenne du nombre de messages après lesquels la mise à jour est faite est de 11. Cette version d'IGLM est très coûteuse en temps d'exécution mais elle raffine la classification dynamique puisqu'elle permet d'améliorer la classification par rapport à la première version et de chercher le bon nombre de messages mal classés après lequel la mise à jour est réalisée.

6.6 Parcimonie ou "Sparsity"

D'une façon générale, une représentation "sparse" (creuse) est définie comme étant une représentation dans laquelle un petit nombre de coefficients contient une proportion large d'énergie [41]. Dans le cadre de la classification des textes courts, la parcimonie représente un problème majeur qui apparaît dans la phase de prétraitement, lors de la construction de la matrice représentative des messages. La matrice

obtenue est généralement composée d'un ensemble de vecteurs avec peu de coefficients non nuls. La diminution de la parcimonie des messages peut être un moyen efficace pour améliorer leur classification.
Nous avons mis en place une série d'expériences pour vérifier si l'enrichissement et l'abstraction, les deux méthodes utilisées pour améliorer la classification des messages, ont bien diminué leur parcimonie.
Pour identifier les mesures utilisées pour calculer la parcimonie, nous avons fait une étude de l'état de l'art de ce sujet. Hurley et al. proposent dans leur article [41] une comparaison entre les mesures existantes de la parcimonie. Cette comparaison est basée sur les 6 propriétés de la parcimonie définies dans le domaine de la finance.
Hurley et al. ont considéré qu'une distribution équitable d'une richesse est une distribution dans laquelle tous les individus ont la même quantité d'énergie, donc la moins "sparse" possible. Ces 6 propriétés sont :

— **Robin Hood (D1)** : permet de diminuer la parcimonie. Son principe est de prendre au riche et donner au pauvre pour diminuer le déséquilibre mais ça ne rend ni le riche pauvre ni le pauvre riche. Mathématiquement ce problème est formulé comme suit :

$$S([c_1...c_i-\alpha...c_j+\alpha...]) < S(\overrightarrow{c}) \quad \alpha, c_i, c_j \textit{ tels que } c_i > c_j \textit{ et } 0 < \alpha < \frac{c_i - c_j}{2} \tag{6.1}$$

— **Scaling (D2)** : la parcimonie pour cette propriété est invariante. Multiplier la richesse avec un facteur constant ne change pas la distribution. Le riche reste riche et le pauvre reste pauvre.

$$S(\alpha\overrightarrow{c}) = S(\overrightarrow{c}),\ \forall\alpha \in \mathbb{R}, \alpha > 0 \tag{6.2}$$

— **Rising tide (D3)** : ajouter une constante suffisamment grande pour chaque individu (un trillion de dollars) diminue la parcimonie. La différence entre les richesses devient négligeable. Dans ce cas, on écarte le cas où tous les individus de départ ont la même richesse car l'ajout de la constante ne change pas la distribution initiale.

$$S(\alpha + \overrightarrow{c}) < S(\overrightarrow{c}),\ \alpha \in \mathbb{R}, \alpha > 0 \tag{6.3}$$

— **Cloning (D4)** : La parcimonie reste la même pour cette propriété. La distribution d'une combinaison des deux populations identiques est la même que la distribution de chacune toute seule.

$$S(\overrightarrow{c}) = S(\overrightarrow{c}||\overrightarrow{c}) = S(\overrightarrow{c}||\overrightarrow{c}||\overrightarrow{c}) = S(\overrightarrow{c}||\overrightarrow{c}||....||\overrightarrow{c}) \tag{6.4}$$

— **Bill gates (P1)** : Quand un seul individu devient infiniment riche, la parcimonie augmente.

$$\forall i \exists \beta > 0, \textit{tel que } \forall\alpha > 0 : S(c_1...c_i + \beta + \alpha...]) > S(c_1...c_i + \beta...) \tag{6.5}$$

Measure	Definition
ℓ^0	$\#\{j, c_j = 0\}$
ℓ^0_ϵ	$\#\{j, c_j \leq \epsilon\}$
$-\ell^1$	$-\left(\sum_j c_j\right)$
$-\ell^p$	$-\left(\sum_j c_j^p\right)^{1/p}, \ 0 < p < 1$
$\frac{\ell^2}{\ell^1}$	$\frac{\sqrt{\sum_j c_j^2}}{\sum_j c_j}$
$-\tanh_{a,b}$	$-\sum_j \tanh\left((ac_j)^b\right)$
$-\log$	$-\sum_j \log\left(1 + c_j^2\right)$
κ_4	$\frac{\sum_j c_j^4}{\left(\sum_j c_j^2\right)^2}$
u_θ	$1 - \min_{i=1,2,\ldots,N-\lceil\theta N\rceil+1} \frac{c_{(i+\lceil\theta N\rceil-1)} - c_{(i)}}{c_{(N)} - c_{(1)}}$ s.t. $\lceil\theta N\rceil \neq N$ for ordered data, $c_{(1)} \leq c_{(2)} \leq \cdots \leq c_{(N)}$
$-\ell^p_-$	$-\sum_{j, c_j \neq 0} c_j^p, \ p < 0$
H_G	$-\sum_j \log c_j^2$
H_S	$-\sum_j \tilde{c_j} \log \tilde{c_j}^2$ where $\tilde{c_j} = \frac{c_j^2}{\lVert\vec{c}\rVert_2^2}$
H'_S	$-\sum_j c_j \log c_j^2$
Hoyer	$(\sqrt{N} - \frac{\sum_j c_j}{\sqrt{\sum_j c_j^2}})(\sqrt{N} - 1)^{-1}$
Gini	$1 - 2\sum_{k=1}^{N} \frac{c_{(k)}}{\lVert\vec{c}\rVert_1} \left(\frac{N-k+\frac{1}{2}}{N}\right)$ for ordered data, $c_{(1)} \leq c_{(2)} \leq \cdots \leq c_{(N)}$

FIGURE 6.3 – les mesures de parcimonie [41].

— **Babies (P2)** : Cette propriété augmente la parcimonie, ajouter des individus avec zéro richesse augmente le déséquilibre de la distribution de la richesse.

$$S(\overrightarrow{c}||0) > S(\overrightarrow{c}) \tag{6.6}$$

Hurley et al. dans [41] ont également résumé l'ensemble des mesures de parcimonie les plus populaires que nous retrouvons dans la figure 6.3.

Pour nos tests nous optons pour les deux mesures suivantes :

— l'indice de Gini : jugé par l'article le plus efficace puisqu'il vérifie les 6 propriétés mentionnées ci-dessus.

— La mesure Hoyer : elle vérifie 5 critères parmi les 6. Elle a échoué à vérifier la propriété D4 : l'invariance contre le clonage, une propriété qui n'a pas beaucoup d'influence sur notre domaine d'étude des messages courts.

Nous avons calculé ces deux mesures pour chaque vecteur de la matrice représentative des textes puis nous avons fait une moyenne sur l'ensemble des mesures obtenues. Cette moyenne représente la mesure de parcimonie générale.
Soit $gini(\overrightarrow{c})$ une fonction qui calcule l'indice de Gini d'un vecteur $\overrightarrow{c}$ et $hoyer(\overrightarrow{c})$ la fonction qui calcule la mesure de Hoyer de ce même vecteur.

La mesure de parcimonie selon l'indice de Gini est alors :

$$parcimonie_{gini} = \frac{1}{m} \times \sum_{i=1}^{m} gini(\overrightarrow{c_i}) \tag{6.7}$$

et la mesure générale de hoyer est :

$$parcimonie_{hoyer} = \frac{1}{m} \times \sum_{i=1}^{m} hoyer(\overrightarrow{c_i}) \tag{6.8}$$

où $\overrightarrow{c_i}$ est le i_{eme} vecteur dans la matrice des textes et m est le nombre de messages composant cette matrice.

A ces deux mesures nous ajoutons une troisième qui est spécifique à notre domaine. Cette mesure est définie par le ratio entre le nombre d'éléments de valeur nulle et le nombre total d'éléments de la matrice.
Soit :

$$parcimonie_{zero} = \frac{1}{m \times n} \times \sum_{i=1}^{n} \sum_{j=1}^{m} \#\{A_{(i,j)} = 0\} \tag{6.9}$$

où A est la matrice représentative des textes, m le nombres de messages et n le nombre de caractéristiques (mots). Le tableau 6.8 résume les résultats obtenus pour ces trois mesures sur les trois méthodes de représentation matricielle.

TABLE 6.8 – Parcimonie pour les trois méthodes de représentation matricielle

	zero number	Gini parcimonie	Hoyer parcimonie
Bag Of Words	0.9994	0.99956	0.9854
Abstraction	0.9993	0.99954	0.9862
Enrichissement	0.9507	0.9662	0.826

A partir de ce tableau nous remarquons que le niveau de parcimonie est très haut dans la matrice construite par le "Bag Of Words" traditionnel : elle est quasiment égale à 1 pour les trois mesures. L'ajout de l'abstraction n'a pas donné de résultats trop significatifs par rapport à ces mesures, nous avons juste observé une très légère diminution. Les résultats sont un peu plus parlants dans le cas de l'enrichissement où la diminution de la parcimonie par les trois mesures par rapport à celle obtenue par la représentation de base est plus significative que celle de l'abstraction. Ceci explique aussi que les résultats de la classification en utilisant l'enrichissement sont meilleurs que ceux obtenus par l'abstraction.

Cependant, même si nous avons réussi à diminuer la parcimonie, cette dernière reste toujours élevée. Ceci s'explique par le fait que même si on réduit le nombres de caractéristiques dans le cas de l'abstraction ou si l'on augmente la taille des textes dans le cas de l'enrichissement, le nombre de mots d'un seul texte reste toujours très petit par rapport au nombre total des mots de tous les textes, d'où la diminution légère de la parcimonie.

6.7 Discrimination des caractéristiques

Comme nous pouvons le voir dans la figure 6.4, la parcimonie représente une vision horizontale pour évaluer la qualité de la représentation matricielle des mes-

sages courts car cette mesure est calculée par message. Un autre moyen permettant d'évaluer la qualité de la matrice mais avec cette fois une vision verticale est l'évaluation de la discrimination de ces caractéristiques.
La discrimination d'une caractéristique est définie par rapport à son gain d'information. En effet plus le gain d'information augmente et se rapproche de 1 plus la caractéristique est discriminante.

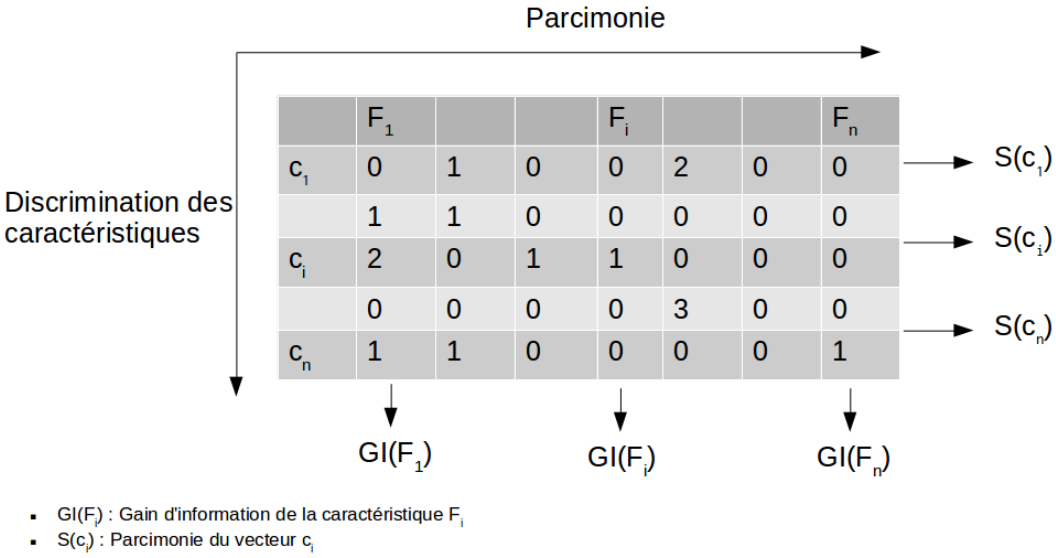


FIGURE 6.4 – Illustration de l'étude de la qualité de la matrice représentative des messages courts.

Normalement, une forêt construite à partir d'une matrice qui contient le plus de caractéristiques discriminantes devrait être la plus performante. Pour vérifier cette hypothèse, nous avons calculé le gain d'information de chaque caractéristique des trois représentations matricielles obtenues par le BOW traditionnel, l'abstraction et l'enrichissement. Le but est de vérifier si l'enrichissement conduit à des caractéristiques meilleures que celles des deux autres méthodes de représentation.

Nous avons ensuite :

1. extrait le gain de la meilleure caractéristique de chacune des trois représentations (critère 1)
2. calculé le ratio des p caractéristiques dont le gain d'information dépasse un seuil que nous avons fixé à 0.05 (critère 2).
3. calculé le ratio des K meilleures caractéristiques dont le gain d'information est supérieur à celui de la moyenne des gains de la représentation traditionnelle (critère 3).

Le tableau 6.9 présente les différents résultats que nous avons obtenus.

En analysant le tableau 6.9, nous remarquons que l'enrichissement dépasse l'abstraction et la représentation traditionnelle en ce qui concerne la qualité des carac-

TABLE 6.9 – Statistiques des gains d'information des trois méthodes de représentation matricielle

	critère 1	critère 2	critère 3
Bag Of Words	0.106	0.4%	50%
Abstraction	0.206	1%	55%
Enrichissement	0.383	11%	73%

téristiques. En effet, l'enrichissement offre le meilleur gain d'information. Beaucoup plus de caractéristiques discriminantes existent dans ce cas que dans le cas de l'abstraction ou celui de la représentation traditionnelle.
Ces résultats prouvent notre hypothèse de départ : l'enrichissement permet la création d'arbres plus performants que ceux construits par l'abstraction car l'enrichissement offre un nombre plus grand de caractéristiques discriminantes.

Cette hypothèse est vérifiée aussi si on compare les résultats de l'abstraction avec ceux de la représentation traditionnelle. On constate en effet que les caractéristiques générées par l'abstraction sont plus discriminantes que celles faites par le BOW. Ceci explique la performance de l'abstraction par rapport au BOW pour la classification des messages.

6.8 Synthèse

Nous avons mené dans ce chapitre une étude comparative des différentes solutions proposées pour améliorer la classification des messages courts qui sont les forêts sémantiques et IGLM. Même si elles sont conçues à partir de deux visions différentes, ces deux approches fournissent deux parties comparables qui sont l'enrichissement et l'abstraction : deux méthodes appliquées dans la phase de prétraitement pour améliorer la qualité des messages courts. Nous avons vu par la suite que ces deux méthodes ont une influence directe sur les résultats des deux approches. Plusieurs axes de comparaison ont été étudiés. L'accuracy, par exemple, a montré un avantage pour l'enrichissement par rapport à l'abstraction, les deux étant par ailleurs meilleures que la représentation "Bag Of Word" traditionnelle (BOW). De plus, l'abstraction est beaucoup plus rapide que l'enrichissement. En se référent à la littérature, l'amélioration de l'accuracy soit par l'enrichissement, soit par l'abstraction par rapport au BOW est due à une diminution de la parcimonie, une caractéristique des messages courts. Nous avons fait un état de l'art du sujet et nous avons identifié les mesures de parcimonie les plus adéquates. Les résultats de l'application de ces mesures sur les différentes approches sont en faveur de l'enrichissement même si le niveau de parcimonie reste toujours élevé. Ceci est expliqué par le fait que même si nous augmentons la taille des messages ou si nous diminuons l'espace des caractéristiques, la différence entre un message et la totalité du vocabulaire reste toujours grande. Nous avons aussi étudié la discrimination des caractéristiques pour chacune de ces deux méthodes. Dans ce cas aussi l'enrichissement est meilleur. Pour

conclure, le choix de la meilleure méthode de prétraitement des messages courts dépend principalement du contexte de travail, l'enrichissement dans notre cas a battu l'abstraction sur plusieurs niveaux, même si les scores restent malgré tout proches. Dans un contexte d'apprentissage en ligne par exemple, l'avantage bascule vers l'abstraction pour privilégier le temps d'exécution.

Enfin, même si les modèles de classification fournis par les deux méthodes ne sont pas comparables en terme d'accuracy, nous avons donné dans ce chapitre une idée sur leurs tailles, sur la complexité des approches ainsi que sur le temps d'exécution nécessaires. Pour résumer, les forêts sémantiques fournissent des arbres beaucoup moins complexes que ceux d'IGLM, la moyenne de leur nombre de nœuds étant beaucoup plus petite. Cependant, l'approche sémantique est moins rapide en terme d'apprentissage car son temps d'exécution est plus important.

Rappel des algorithmes :

Algorithme 4 : Algorithme d'enrichissement

Entrée: Ensemble de messages courts, ensemble de thèmes
Sortie: Ensemble de messages enrichis

```
POUR chaque message dans l'ensemble des messages :
    /* enrichissement niveau mot */
    POUR chaque mot dans message :
        POUR chaque thème dans l'ensemble des thèmes :
            SI le mot appartient au thème :
                poidsThème [thème] ← poidsDansThème[mot]
            SINON :
                poidsThème [thème] ← 0
        meilleurThème ← trouverMaximum(poidsThème)
      thèmesAAjouterNiveauMot ← thèmesAAjouterNiveauMot + meilleurThème

    enrichir(message, thèmesAAjouterNiveauMot)
    / * enrichissement niveau message */
similarités ← [ ]
meilleurThème ← [ ]
POUR chaque thème dans l'ensemble des thèmes :
        similarité ← calculerNombreDeMotsEnCommun (messages, thème)
        ajouterSimilarité(similarités,similarité)
        meilleursThèmes ← trouverPMeilleursthèmes(similarités)
POUR chaque thème dans meilleursThèmes :
        enrichir(message,thème)
```

Algorithme 5 : Arbres sémantiques

Entrée: Car[N] : Ensemble de caractéristiques construites à partir des messages enrichis,
N : nombre total de caractéristiques,
L : nombre de caractéristiques choisies aléatoirement,
Seuil : poids minimum des mots considérés,
Sortie: Forêt sémantique

```
\* Construction du réseau sémantique *\
réseau = appliquerLDASur(car[N])
POUR chaque arbre :
    espaceInitial = choisir aléatoirement L caractéristiques < N
    /* l'élargissement de l'espace de caractéristiques */
    espaceCaracteristiquesFinal ← [ ]
    POUR chaque caractéristique dans espaceInitial :
    /* chercher le thème le plus proche */
    ListeCaracteristiquesAAjouter = SFS(caractéristique,réseau, Seuil)
    SI ListeCaracteristiquesAAjouter <> [ ]
        espaceCaracteristiquesFinal += ListeCaracteristiquesAAjouter
    Construire l'arbre en tenant compte seulement d'espaceCaracteristiquesFinal
```

Algorithme 6 : Semantic Feature Selection (SFS)

Entrée: caractéristique, réseau, seuil
Sortie: ListeDeCaractéristiques[ ]

```
thème = chercher le thème le plus proche (caractéristique, réseau)
Si thème <>[ ]
    POUR chaque mot dans thème
        Si Poids(mot) > seuil
            ListeDeCaractéristiques + = mot
    retourner ( ListeDeCaractéristiques)
SINON
    retourner [ ]
```

Algorithme 7 : IGLM

Entrée: base de messages, ensemble de thèmes, delta, nombre d'arbres
Sortie: forêt aléatoire interactive

```
/* Abstraction des caractéristiques/
POUR chaque message dans base de messages
    POUR chaque mot dans message
        trouver le thème le plus proche(mot, ensemble de thèmes)
        SI thème existe
            Remplacer le mot par le mot représentant le thème
/* Première construction de forêt aléatoire */
RF = créer première forêt(messages abstraits, nombre d'arbres)
/* Calculer la condition de mise à jour */
SomGain1 = 0
Pour F_i dans F /* F = espace de caractéristique */
    Gain(F_i) = calculer le gain d'information(F_i)
    SomGain1+ = Gain(F_i)
ε = sqrt(R^2.ln(1/δ) / 2N)
/* Mise à jour de la forêt aléatoire */
classe = prédire classe (RF, nouveau message)
SI classe n'est pas correcte
    base de messages += nouveau message
    SomGain2 = 0
    Pour F_i dans F
        Gain(F_i) = calculer gain d'information(F_i)
        SomGain2+ = Gain(F_i)
        if |SomGain2 − SomGain1| > ε
        /* Adaptation */
        calculer l'accuracy pour chaque arbre
        z = 0.1 × nombre total d'arbres
        trouver les z arbres les plus mauvais
        créer z nouveaux arbres
        remplacer les z arbres identifiés par les z nouveaux.
        mettre à jour ε
        SomGain1 = SomGain2
```

Chapitre 7

Conclusion générale

7.1 Conclusion

Cette thèse s'inscrit dans le cadre de la classification automatique des messages courts. Son but majeur est d'améliorer cette classification pour diminuer les corrections manuelles des erreurs de classification. Nous avons proposé ainsi deux approches qui traitent le sujet sous deux angles différents pour répondre à deux objectifs distincts qui sont :

— l'introduction de la sémantique en identifiant les différents liens à la fois existants entre les messages et non visibles en considérant leur représentation traditionnelle ;

— la mise en place d'un mécanisme interactif qui profite de l'expertise de l'utilisateur pour améliorer la qualité du modèle de classification au cours du temps.

Pour répondre à notre premier objectif, nous avons proposé les forêts sémantiques, qui est une approche statique composée de deux parties majeures. La première partie agit au niveau prétraitement du processus de la classification des messages courts. Elle a pour but d'améliorer leur mauvaise qualité due principalement à la parcimonie et au manque de contexte caractérisant ce type de données. La méthode consiste à enrichir les messages courts en utilisant de la connaissance externe. Pour construire la source d'enrichissement, nous avons utilisé une approche bien connue dans la littérature qui applique LDA, une méthode des "Topic Model", sur un ensemble de documents externes sémantiquement proches de l'ensemble des messages courts et couvrant suffisamment leurs domaines. Le résultat de cette étape est un ensemble de thèmes où chacun est constitué d'un ensemble de mots qui sont sémantiquement liés. Un poids est associé à chaque mot appartenant à un thème. Il représente son importance dans ce thème. Sur cet ensemble de thèmes représentant notre source d'enrichissement, nous définissons des mesures de similarités qui vont nous permettre d'étendre chaque message par les thèmes qui lui sont les plus proches. Nous cherchons dans un premier temps pour chaque mot de chaque message le thème dans lequel ce mot possède le poids le plus fort, puis, nous lui ajoutons tous les mots constituant ce thème. Ce premier niveau d'enrichissement permettra ainsi de transformer un message court constitué d'un simple ensemble de mots en un ensemble de concepts généraux permettant de couvrir le plus possible les domaines auxquels il est lié. Ce premier niveau d'enrichissement est suivi par un autre qui considère la sémantique de la globalité du message. Un sous ensemble de thèmes ayant le nombre de mots en commun le plus grand avec le message en

question lui sont ajoutés à chaque message. Ce deuxième niveau d'enrichissement permet de mieux remplir les lacunes sémantiques entre les messages et de résoudre quelques problèmes d'ambiguïté sémantique, notamment la polysémie.

Nous avons associé à cette méthode d'enrichissement un nouveau type de forêts aléatoires qui introduit la sémantique dans leur comportement. Nous avons ainsi proposé une première version que nous avons appelée arbres sémantiques. Cette implémentation utilise un réseau, préalablement construit sur l'ensemble de caractéristiques obtenues après la phase d'enrichissement et offrant ainsi des connexions sémantiques entre elles. Nous utilisons LDA une deuxième fois pour construire ce réseau. Un arbre sémantique est ainsi construit en se basant sur un ensemble de caractéristiques sémantiquement liées. Nous choisissons initialement et aléatoirement un sous ensemble de caractéristiques permettant de conserver le principe de diversité dans la construction des forêts aléatoires. Nous élargissons ensuite cet espace en ajoutant à chaque caractéristique les mots qui lui sont sémantiquement les plus proches. Cette procédure s'appelle "Semantic Feature Selection". Une fois que l'espace des caractéristiques pour construire un arbre est identifié, nous appliquons la technique classique de "Random Feature Selection" sur cet espace pour construire chaque nœud. Pour éviter un éventuel risque de corrélation, nous avons déplacé la procédure "Semantic Feature Selection" au niveau nœud. Ainsi un arbre est construit en tenant compte de toutes les caractéristiques. Cependant, un nœud est construit à partir d'un ensemble qui n'est pas choisi de façon complètement aléatoire mais dont les éléments ont des liens sémantiques.

Pour valider notre approche nous l'avons testée sur le benchmarck "search snippets" qui est bien connu dans la littérature de la classification des textes courts. Nous avons varié les tests afin de mesurer la valeur ajoutée de chaque étape dans l'approche. Notre méthode a amélioré la classification des "snippets" par rapport aux forêts aléatoires traditionnelles quelque soit le nombre d'arbres utilisés. Elle a fourni également des arbres moins complexes que ceux fournis par les forêts traditionnelles. Le nombre de nœuds des arbres sémantiques est deux fois plus petit que le nombre de nœuds des arbres traditionnels. Des tests statistiques ont également confirmé ces résultats.

Notre deuxième contribution dans cette thèse est la méthode IGLM (Interactive Generic Learning Method). C'est un processus dynamique inspiré de l'apprentissage actif et de l'apprentissage en ligne qui prend en considération les caractéristiques des textes courts. L'idée globale est de profiter de l'intervention de l'utilisateur pour améliorer la qualité de la classification en mettant à jour le modèle par les nouvelles données arrivant au cours du temps et sur lesquelles le modèle courant s'est trompé de classe. IGLM est ainsi décomposée en trois grandes parties :

— une méthode d'abstraction dans la phase de prétraitement permettant d'améliorer la qualité des messages courts. En effet, elle réduit la taille de la matrice des données afin de condenser l'information dans cette dernière et d'atténuer l'effet de la parcimonie. La méthode consiste à créer un ensemble d'abstractions où chaque abstraction représente un regroupement sémantique de plusieurs caractéristiques similaires. Une fois les abstractions créées, la mé-

thode les introduit par la suite dans les messages en remplaçant chaque mot par son abstraction si un tel lien entre eux existe. Les mots ayant une même abstraction ne sont plus considérés comme deux caractéristiques distinctes mais plutôt comme une seule. Une fois tous les messages abstraits, la matrice représentative des données est générée donnant suite à la création d'une forêt aléatoire initiale.

— condition de mise à jour du modèle : cette deuxième étape se répète au cours du temps. Elle est déclenchée à chaque fois qu'un nouveau message est mal classé par la forêt actuelle. Le but est de déterminer le bon moment pour mettre à jour le modèle, en d'autre termes après combien de messages mal classés, faut-il déclencher la mise à jour ? Nous utilisons un résultat statistique très connu dans le domaine de l'apprentissage en ligne qui est l'inégalité de Hoeffding. Nous calculons les gains d'information de toutes les caractéristiques avant et après l'arrivée des nouveaux messages et nous les comparons. Si un minimum d'écart défini par Hoeffding est détecté nous mettons alors le modèle à jour.

— l'adaptation : c'est le mécanisme de mise à jour qui consiste à écarter un sous ensemble d'arbres identifiés comme les plus mauvais de la forêt actuelle et à les remplacer par d'autres construit sur la base de la nouvelle représentation.

Dans le but d'évaluer cette méthode, nous avons établi le protocole expérimental suivant : nous avons ainsi réorganisé le corpus "search snippets" pour simuler des données arrivant au cours du temps et qui servent à mettre à jour le modèle de classification. Nous avons ensuite varié les tests afin de déterminer la contribution de chaque partie. Nous avons tracé l'évolution de l'accuracy qui augmente au fur et à mesure des mises à jours. Nous avons également démontré que l'utilisation de Hoeffding est plus judicieuse que de fixer le nombre de messages à partir duquel le modèle se met à jour.

Nous avons fini cette thèse par une étude comparative des différentes techniques employées dans nos deux approches sur plusieurs axes tels que la taille des forêts, le temps de calcul, la complexité et la nature des données. Nous avons vu que plusieurs points communs existent entre ces techniques et elles diffèrent en d'autres. En effet, selon le contexte et la nature de l'application, une méthode peut être plus adaptée qu'une autre.

7.2 Perspectives

Les résultats obtenus lors de l'évaluation des forêts sémantiques sont fortement liés à la qualité de la base d'enrichissement. Celle-ci doit être bien représentative de l'ensemble de messages à enrichir. Une base trop générale peut introduire du bruit, en contrepartie, une base réduite n'assure pas un maximum d'enrichissement. Il a été démontré dans plusieurs travaux que LDA est une technique adaptée à notre domaine de recherche, ce qui nous a encouragé à la privilégier par rapport à d'autres méthodes de "Topic Models" et à l'employer deux fois dans notre approche. Cepen-

dant l'approche reste flexible : LDA peut être facilement remplacée par n'importe quelle autre technique de construction de la source d'enrichissement et du réseau sémantique des caractéristiques. Par exemple, GPU_DMM est une nouvelle technique des "Topic Models" qui a été récemment proposée par Li et al. dans [49] et qui est spécifique aux textes courts. Cette technique se base sur le plongement de mots ou "word embedding" qui fait partie de l'apprentissage profond ("deep learning") et qui opte pour une représentation vectorielle des mots et des textes dans un espace continu au lieu d'une représentation traditionnelle. GPU_DMM semble très intéressante grâce à sa spécificité aux messages courts et grâce aussi aux résultats présentés dans [49]. Nous estimons ainsi que sa combinaison avec les forêts sémantiques peut améliorer davantage la classification des messages courts. Notre approche est également compatible avec les forêts aléatoires pondérées. Nous estimons aussi que définir une heuristique permettant d'attribuer à chaque classe un poids inversement proportionnel à sa représentativité dans la base d'apprentissage permettra d'améliorer la classification des benchmarks déséquilibrés tels que "search snippets".

Nous avons démontré dans IGLM que l'utilisation de l'inégalité de Hoeffding est plus intéressante que l'utilisation d'un simple paramètre fixé par l'utilisateur. Cependant, elle est plus coûteuse en terme de temps à cause du calcul répétitif des gains d'information de toutes les caractéristiques après l'ajout de chaque message. Afin d'optimiser le temps de calcul, Domingos et al. affirment dans [28] qu'il est très peu probable qu'un seul message soit assez informatif pour déclencher une mise à jour, il est donc plus intéressant de définir un nombre de messages à partir duquel le calcul de gains d'information est lancé, ce qui permettrait d'économiser du temps. La mise à jour du modèle de classification dans IGLM consiste à remplacer les mauvais classifieurs par de nouveaux plus performants. Nous pensons qu'il est possible de construire une nouvelle méthode hybride dans laquelle plusieurs types de classifieurs coexisteraient. Cela permettrait en effet d'apporter plus de diversité dans le processus de vote et de dépasser les limites posées par l'utilisation d'un seul type de classifieurs.

Pour la classification des messages courts, comme nous l'avons déjà dit, dernièrement, la tendance au niveau des recherches en fouille de données se dirige vers l'apprentissage profond, il a été appliqué avec succès dans plusieurs domaines dont notamment les textes courts avec la technique du "word embedding". Cette technique permet en effet de détecter les liens sémantiques entre les textes. Kenter et al. dans [42] ont utilisé cette technique pour calculer la similarité entre les textes courts. Nous pouvons ainsi nous inspirer de ces travaux pour améliorer davantage nos mesures de similarité présentées dans nos différents travaux.

De point de vue classification dynamique des messages, récemment, un nouvel algorithme d'apprentissage en ligne est proposé dans [48]. Il s'agit de "Mondrian Forests" qui est un ensemble d'arbres de décisions construits en se basant sur les processus de Mondrian [65]. Par rapport aux algorithmes existants, cet algorithme se distingue par sa capacité à apprendre un modèle de classification en utilisant un nombre d'exemples plus réduit, Nous pensons que cet algorithme peut avoir

des propriétés intéressantes qui peuvent améliorer la classification dynamique des messages courts.

7.3 Conclusion et perspectives pour "Semantic Grouping Company"

Cette thèse s'est déroulée dans le cadre d'une collaboration entre le laboratoire I3S et l'entreprise "Semantic Grouping Company" (SGC). SGC développe le logiciel "Meeting Software" (MS) dont le but est d'améliorer la qualité de séminaires d'entreprises. Pour cela MS traite des messages courts pour favoriser la production d'une vision collective de l'entreprise et de ses évolutions. En effet, ces messages sont issus généralement de réponses à des questions posées en ligne au cours du séminaire. De par leur nature, les données traitées dans MS présentent plusieurs défis liés à la parcimonie du contenu et à l'absence de contexte. Nos travaux ont été intégrés dans MS au fur et à mesure de leur avancement. La dernière version de la plate-forme développée par SGC au cours des derniers mois de la thèse a permis de mieux structurer mes contributions et de les intégrer dans MS. En plus de cette plate-forme, SGC nous a également régulièrement fourni des données pour évaluer nos contributions, mais ce sont plus particulièrement les benchmarks les plus complets issus de plusieurs séminaires transmis en fin de thèse qui ont permis une évaluation approfondie des différentes solutions que j'ai pu proposer. Nous avons ainsi développé une nouvelle méthode hybride de classification, une méthode collaborative et dynamique qui met en concurrence plusieurs types de classifieurs afin de trouver le plus adéquat pour la classification de chaque message.

Plusieurs pistes d'amélioration restent à explorer dans MS. Au niveau du prétraitement des données, les nouvelles recherches se dirigent vers l'apprentissage profond permettant ainsi de fournir une nouvelle représentation de données où la sémantique des messages est mieux identifiée. Vue les performances remarquables démontrées dans la littérature de l'apprentissage profond, nous pensons que son application sur les données de SGC peut être très avantageuse pour réduire le niveau très élevé de parcimonie, permettant ainsi aux algorithmes d'apprentissage de mieux apprendre les modèles de classification. Dans le contexte de la classification des textes, l'apprentissage profond a par exemple été appliqué avec succès pour les textes courts avec des techniques de "word embeddings". Ces techniques permettent en effet de détecter les liens sémantiques entre les textes. Nous pouvons ainsi l'employer dans nos mesures de similarité afin de les améliorer davantage lors de la phase de prétraitement.

Bibliographie

[1] Hanady Abdulsalam, David B Skillicorn, and Patrick Martin. Streaming random forests. In *Database Engineering and Applications Symposium, 2007. IDEAS 2007. 11th International*, pages 225–232. IEEE, 2007. (Cité en pages vii, 56, 57, 60 et 88.)

[2] Rakesh Agrawal, Sakti Ghosh, Tomasz Imielinski, Bala Iyer, and Arun Swami. An interval classi er for database mining applications. In *Proc. of the VLDB Conference*, pages 560–573, 1992. (Cité en page 61.)

[3] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 9(7) :1545–1588, 1997. (Cité en page 51.)

[4] Ion Androutsopoulos, John Koutsias, Konstantinos V Chandrinos, George Paliouras, and Constantine D Spyropoulos. An evaluation of naive bayesian anti-spam filtering. *arXiv preprint cs/0006013*, 2000. (Cité en page 37.)

[5] Laurent Miclet Antoine cornuéjols. *Apprentissage artificiel concepts et algorithmes*. EYROLLES, 2002. (Cité en pages vii et 34.)

[6] R Arun, Venkatasubramaniyan Suresh, CE Veni Madhavan, and MN Narasimha Murthy. On finding the natural number of topics with latent dirichlet allocation : Some observations. In *Advances in Knowledge Discovery and Data Mining*, pages 391–402. Springer, 2010. (Cité en page 14.)

[7] Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1) :39–71, 1996. (Cité en pages 7, 35, 43 et 78.)

[8] Simon Bernard. *Forêts Aléatoires : De l'Analyse des Mécanismes de Fonctionnement à la Construction Dynamique*. PhD thesis, Université de Rouen, 2009. (Cité en page 55.)

[9] Simon Bernard, Sébastien Adam, and Laurent Heutte. Dynamic random forests. *Pattern Recognition Letters*, 33(12) :1580–1586, 2012. (Cité en page 56.)

[10] Simon Bernard, Laurent Heutte, and Sébastien Adam. Forest-rk : A new random forest induction method. In *International Conference on Intelligent Computing*, pages 430–437. Springer, 2008. (Cité en page 51.)

[11] Gérard Biau, Luc Devroye, and Gábor Lugosi. Consistency of random forests and other averaging classifiers. *The Journal of Machine Learning Research*, 9 :2015–2033, 2008. (Cité en page 54.)

[12] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa : Massive online analysis. *The Journal of Machine Learning Research*, 11 :1601–1604, 2010. (Cité en page 61.)

[13] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3 :993–1022, 2003. (Cité en pages vii, 8, 12, 13, 14, 20, 29, 35, 37, 42, 65, 72 et 90.)

[14] Leo Breiman. Bagging predictors. *Machine learning*, 24(2) :123–140, 1996. (Cité en pages 45, 50 et 59.)

[15] Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001. (Cité en pages 45, 49, 52, 55, 61, 64, 72, 73 et 104.)

[16] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984. (Cité en pages 45, 48 et 88.)

[17] Wray Buntine. Learning classification trees. *Statistics and computing*, 2(2) :63–73, 1992. (Cité en page 58.)

[18] Cornelia Caragea, Nathan McNeese, Anuj Jaiswal, Greg Traylor, Hyun Woo Kim, Prasenjit Mitra, Dinghao Wu, Andrea H Tapia, Lee Giles, Bernard Jansen, et al. Classifying text messages for the haiti earthquake. In *Information Systems for Crisis Response and Management, ISCRAM*, 2011. (Cité en pages 27, 29, 31, 35, 38, 41, 42 et 43.)

[19] Doina Caragea, Vikas Bahirwani, Waleed Aljandal, and William H Hsu. Ontology-based link prediction in the livejournal social network. In *SARA*, volume 9, pages 1–1, 2009. (Cité en pages 71 et 72.)

[20] Jason Catlett. *Megainduction : machine learning on very large databases*. Basser Department of Computer Science, University of Sydney, 1991. (Cité en page 58.)

[21] Mengen Chen, Xiaoming Jin, and Dou Shen. Short text classification improved by learning multi-granularity topics. In *IJCAI*, pages 1776–1781, 2011. (Cité en pages vii, 8, 21, 22, 28, 35, 36, 40, 41 et 42.)

[22] Zheng Chen and Weixiong Zhang. Integrative analysis using module-guided random forests reveals correlated genetic factors related to mouse weight. *PLoS computational biology*, 9(3) :e1002956, 2013. (Cité en page 55.)

[23] Saïp Ciss. *Forêts uniformément aléatoires et détection des irrégularités aux cotisations sociales*. PhD thesis, Paris 10, 2014. (Cité en page 53.)

[24] William W Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *ACM SIGMOD Record*, volume 27, pages 201–212. ACM, 1998. (Cité en page 39.)

[25] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3) :273–297, 1995. (Cité en pages 7, 33, 43, 78 et 88.)

[26] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6) :391, 1990. (Cité en pages vii, 8, 9, 10, 42 et 46.)

[27] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977. (Cité en page 11.)

[28] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge*

discovery and data mining, pages 71–80. ACM, 2000. (Cité en pages 57, 59, 60, 61, 88, 92, 93 et 126.)

[29] Jerry Alan Fails and Dan R Olsen Jr. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45. ACM, 2003. (Cité en page 88.)

[30] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 329–338. ACM, 2009. (Cité en page 61.)

[31] Edmund A Gehan. A generalized wilcoxon test for comparing arbitrarily singly-censored samples. *Biometrika*, 52(1-2) :203–223, 1965. (Cité en page 84.)

[32] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1) :3–42, 2006. (Cité en pages 55 et 71.)

[33] Helmut Grabner and Horst Bischof. On-line boosting and vision. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 260–267. IEEE, 2006. (Cité en page 60.)

[34] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301) :13–30, 1963. (Cité en pages 58 et 92.)

[35] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999. (Cité en pages vii, 11 et 42.)

[36] Thomas Hofmann, Jan Puzicha, and Michael I Jordan. Learning from dyadic data. *Advances in neural information processing systems*, pages 466–472, 1999. (Cité en pages 8, 11 et 13.)

[37] Andreas Hotho, Alexander Maedche, and Steffen Staab. Text clustering based on good aggregations. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 607–608. IEEE, 2001. (Cité en page 15.)

[38] 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. (Cité en page 15.)

[39] Xia Hu, Nan Sun, Chao Zhang, and Tat-Seng Chua. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 919–928. ACM, 2009. (Cité en pages vii, 7, 8, 18, 19 et 42.)

[40] Xiaohua Hu, Xiaodan Zhang, Caimei Lu, Eun K Park, and Xiaohua Zhou. Exploiting wikipedia as external knowledge for document clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 389–396. ACM, 2009. (Cité en pages vii, 15, 16, 17, 18, 28 et 42.)

[41] Niall Hurley and Scott Rickard. Comparing measures of sparsity. *Information Theory, IEEE Transactions on*, 55(10) :4723–4741, 2009. (Cité en pages viii, 113, 114 et 115.)

[42] Tom Kenter and Maarten de Rijke. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1411–1420. ACM, 2015. (Cité en page 126.)

[43] Kenji Kira and Larry A Rendell. The feature selection problem : Traditional methods and a new algorithm. In *AAAI*, volume 2, pages 129–134, 1992. (Cité en pages vii, 31, 32 et 43.)

[44] Myungsook Klassen and Nikhila Paturi. Web document classification by keywords using random forests. In *International Conference on Networked Digital Technologies*, pages 256–261. Springer, 2010. (Cité en page 62.)

[45] Ron Kohavi and Clayton Kunz. Option decision trees with majority votes. In *ICML*, volume 97, pages 161–169. Citeseer, 1997. (Cité en page 58.)

[46] Igor Kononenko. Estimating attributes : analysis and extensions of relief. In *Machine Learning : ECML-94*, pages 171–182. Springer, 1994. (Cité en page 22.)

[47] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1) :79–86, 1951. (Cité en pages 12 et 14.)

[48] Balaji Lakshminarayanan, Daniel M Roy, and Yee Whye Teh. Mondrian forests : Efficient online random forests. In *Advances in neural information processing systems*, pages 3140–3148, 2014. (Cité en page 126.)

[49] Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. Topic modeling for short texts with auxiliary word embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 165–174. ACM, 2016. (Cité en page 126.)

[50] Huan Liu and Hiroshi Motoda. *Feature extraction, construction and selection : A data mining perspective*. Springer Science & Business Media, 1998. (Cité en page 25.)

[51] Oded Maron and Andrew W Moore. Hoeffding races : Accelerating model selection search for classification and function approximation. *Advances in neural information processing systems*, pages 59–59, 1994. (Cité en pages 58 et 92.)

[52] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes-which naive bayes ? In *CEAS*, volume 17, pages 28–69, 2006. (Cité en page 37.)

[53] George A Miller. Wordnet : a lexical database for english. *Communications of the ACM*, 38(11) :39–41, 1995. (Cité en page 15.)

[54] Carl N Morris. Parametric empirical bayes inference : theory and applications. *Journal of the American Statistical Association*, 78(381) :47–55, 1983. (Cité en page 13.)

[55] Nikunj C Oza. Online bagging and boosting. In *Systems, man and cybernetics, 2005 IEEE international conference on*, volume 3, pages 2340–2345. IEEE, 2005. (Cité en pages 59 et 88.)

[56] Jussi Pakkanen, Jukka Iivarinen, and Erkki Oja. The evolving tree ?a novel self-organizing network for data analysis. *Neural Processing Letters*, 20(3) :199–211, 2004. (Cité en page 60.)

[57] Christos H Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing : A probabilistic analysis. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 159–168. ACM, 1998. (Cité en page 8.)

[58] Bernhard Pfahringer, Geoffrey Holmes, and Richard Kirkby. New options for hoeffding trees. In *AI 2007 : Advances in Artificial Intelligence*, pages 90–99. Springer, 2007. (Cité en pages vii, 58 et 61.)

[59] 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. (Cité en pages vii, 7, 8, 19, 20, 22, 27, 28, 35, 36, 40, 41, 42, 65, 74, 94 et 104.)

[60] Helena Sofia Pinto and João P Martins. Ontologies : how can they be built ? *Knowledge and information systems*, 6(4) :441–464, 2004. (Cité en page 15.)

[61] J Ross Quinlan. C4. 5 : Programming for machine learning. *Morgan Kauffmann*, 38, 1993. (Cité en page 48.)

[62] J Ross Quinlan et al. Bagging, boosting, and c4. 5. In *AAAI/IAAI, Vol. 1*, pages 725–730, 1996. (Cité en page 56.)

[63] PC Rafeeque and S Sendhilkumar. A survey on short text analysis in web. In *Advanced Computing (ICoAC), 2011 Third International Conference on Advanced Computing*, pages 365–371. IEEE, 2011. (Cité en page 8.)

[64] Juan Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, 2003. (Cité en pages 17 et 24.)

[65] Daniel M Roy and Yee W Teh. The mondrian process. In *Advances in neural information processing systems*, pages 1377–1384, 2009. (Cité en page 126.)

[66] Amir Saffari, Christian Leistner, Jakob Santner, Martin Godec, and Horst Bischof. On-line random forests. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1393–1400. IEEE, 2009. (Cité en pages 59 et 88.)

[67] Tara N Sainath, Sameer Maskey, Dimitri Kanevsky, Bhuvana Ramabhadran, David Nahamoo, and Julia Hirschberg. Sparse representations for text categorization. In *INTERSPEECH*, volume 10, pages 2266–2269, 2010. (Cité en page 7.)

[68] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–278. ACM, 2002. (Cité en page 88.)

[69] Karl-Michael Schneider. Techniques for improving the performance of naive bayes for text classification. In *Computational Linguistics and Intelligent Text Processing*, pages 682–693. Springer, 2005. (Cité en pages 37, 78 et 88.)

[70] Catarina Silva and Bernardete Ribeiro. On text-based mining with active learning and background knowledge using svm. *Soft Computing*, 11(6) :519–530, 2007. (Cité en page 88.)

[71] Catarina Silva and Bernardete Ribeiro. Improving text classification performance with incremental background knowledge. In *International Conference on Artificial Neural Networks*, pages 923–931. Springer, 2009. (Cité en page 88.)

[72] Adrian Silvescu, Cornelia Caragea, and Vasant Honavar. Combining superstructuring and abstraction on sequence classification. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 986–991. IEEE, 2009. (Cité en pages vii, 25, 26, 27 et 43.)

[73] Yangqiu Song, Haixun Wang, Zhongyuan Wang, Hongsong Li, and Weizhu Chen. Short text conceptualization using a probabilistic knowledgebase. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2330–2336. AAAI Press, 2011. (Cité en page 8.)

[74] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842. ACM, 2010. (Cité en pages 7, 8, 32, 37, 40 et 43.)

[75] W Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382. ACM, 2001. (Cité en page 61.)

[76] Aixin Sun. Short text classification using very few words. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 1145–1146. ACM, 2012. (Cité en pages vii, 7, 32, 37, 38, 39 et 43.)

[77] Jiliang Tang, Xufei Wang, Huiji Gao, Xia Hu, and Huan Liu. Enriching short text representation in microblog for clustering. *Frontiers of Computer Science*, 6(1) :88–101, 2012. (Cité en pages vii, 7, 8, 22, 23 et 24.)

[78] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12) :2544–2558, 2010. (Cité en page 8.)

[79] 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. (Cité en page 87.)

[80] Bing-kun Wang, Yong-feng Huang, Wan-xia Yang, and Xing Li. Short text classification based on strong feature thesaurus. *Journal of Zhejiang University SCIENCE C*, 13(9) :649–659, 2012. (Cité en page 37.)

[81] Fang Wang, Zhongyuan Wang, Zhoujun Li, and Ji-Rong Wen. Concept-based short text classification and ranking. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1069–1078. ACM, 2014. (Cité en page 38.)

[82] Xiang Wang, Ruhua Chen, Yan Jia, and Bin Zhou. Short text classification using wikipedia concept based document representation. In *Information Technology and Applications (ITA), 2013 International Conference on*, pages 471–474. IEEE, 2013. (Cité en pages 7, 8 et 35.)

[83] Ben James Winer, Donald R Brown, and Kenneth M Michels. *Statistical principles in experimental design*, volume 2. McGraw-Hill New York, 1971. (Cité en page 84.)

[84] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. Probase : A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 481–492. ACM, 2012. (Cité en page 38.)

[85] Lili Yang, Chunping Li, Qiang Ding, and Li Li. Combining lexical and semantic features for short text classification. *Procedia Computer Science*, 22 :78–86, 2013. (Cité en pages 7, 27, 35, 40, 42 et 43.)

[86] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997. (Cité en pages 30 et 90.)

[87] Illhoi Yoo, Xiaohua Hu, and Il-Yeol Song. Integration of semantic-based bipartite graph representation and mutual refinement strategy for biomedical literature clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 791–796. ACM, 2006. (Cité en page 15.)

[88] Sarah Zelikovitz and H Hirsh. Transductive lsi for short text classification problems. In *FLAIRS conference*, pages 556–561, 2004. (Cité en pages 11, 28, 43 et 63.)

[89] Sarah Zelikovitz and Haym Hirsh. Improving short text classification using unlabeled background knowledge to assess document similarity. In *Proceedings of the seventeenth international conference on machine learning*, volume 2000, pages 1183–1190, 2000. (Cité en page 39.)

[90] Xiaodan Zhang, Liping Jing, Xiaohua Hu, Michael Ng, and Xiaohua Zhou. A comparative study of ontology based term similarity measures on pubmed

document clustering. In *Advances in Databases : Concepts, Systems and Applications*, pages 115–126. Springer, 2007. (Cité en page 15.)

[91] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model : a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4) :43–52, 2010. (Cité en page 6.)

[92] Ying Zhao and George Karypis. Comparison of agglomerative and partitional document clustering algorithms. Technical report, DTIC Document, 2002. (Cité en page 16.)

Résumé : La classification automatique des messages courts est de plus en plus employée de nos jours dans diverses applications telles que l'analyse des sentiments ou la détection des "spams". Par rapport aux textes traditionnels, les messages courts, comme les tweets et les SMS, posent de nouveaux défis à cause de leur courte taille, leur parcimonie et leur manque de contexte, ce qui rend leur classification plus difficile. Nous présentons dans cette thèse deux nouvelles approches visant à améliorer la classification de ce type de message. Notre première approche est nommée «forêts sémantiques». Dans le but d'améliorer la qualité des messages, cette approche les enrichit à partir d'une source externe construite au préalable. Puis, pour apprendre un modèle de classification, contrairement à ce qui est traditionnellement utilisé, nous proposons un nouvel algorithme d'apprentissage qui tient compte de la sémantique dans le processus d'induction des forêts aléatoires. Notre deuxième contribution est nommée «IGLM » (Interactive Generic Learning Method). C'est une méthode interactive qui met récursivement à jour les forêts en tenant compte des nouvelles données arrivant au cours du temps, et de l'expertise de l'utilisateur qui corrige les erreurs de classification. L'ensemble de ce mécanisme est renforcé par l'utilisation d'une méthode d'abstraction permettant d'améliorer la qualité des messages. Les différentes expérimentations menées en utilisant ces deux méthodes ont permis de montrer leur efficacité. Enfin, la dernière partie de la thèse est consacrée à une étude complète et argumentée de ces deux prenant en compte des critères variés tels que l'accuracy, la rapidité, etc.

Mots clés : Classification des messages courts, Sémantique, Forêts aléatoires, Interactivité.

Interactive Learning Methods for short text classification

Abstract :

Automatic short text classification is more and more used nowadays in various applications like sentiment analysis or spam detection. Short texts like tweets or SMS are more challenging than traditional texts. Therefore, their classification is more difficult owing to their shortness, sparsity and lack of contextual information. We present two new approaches to improve short text classification. Our first approach is "Semantic Forest". The first step of this approach proposes a new enrichment method that uses an external source of enrichment built in advance. The idea is to transform a short text from few words to a larger text containing more information in order to improve its quality before building the classification model. Contrarily to the methods proposed in the literature, the second step of our approach does not use traditional learning algorithm but proposes a new one based on the semantic links among words in the Random Forest classifier. Our second contribution is "IGLM" (Interactive Generic Learning Method). It is a new interactive approach that recursively updates the classification model by considering the new data arriving over time and by leveraging the user intervention to correct misclassified data. An abstraction method is then combined with the update mechanism to improve short text quality. The experiments performed on these two methods show their efficiency and how they outperform traditional algorithms in short text classification. Finally, the last part of the thesis concerns a complete and argued comparative study of the two proposed methods taking into account various criteria such as accuracy, speed, etc.

Keywords : Short text classification, Semantics, Random Forest, Interactivity.