République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Ferhat Abbas - Sétif -1

Thèse

Présentée à la Faculté des Sciences

Département Informatique

Pour l'Obtention du Diplôme de

DOCTORAT EN SCIENCES

Option : INFORMATIQUE

Par :

Mouhoub BELAZZOUG

Thème

Apprentissage statistique pour l'extraction des relations à partir de textes

Soutenu le : 06 /07/2021 devant le jury composé de :

Président	Abdelhak BOUBETRA	Professeur	Université de Sétif -1-
Directeur de thèse	Abdallah KHABABA	Professeur	Université de BBA
Examinateur	Samir AKHROUF	Professeur	Université de M'SILA.
Examinateur	Sadik BESSOU	M.C.A	Université de Sétif -1-
Examinateur	Mohamed REDJIMI	M.C.A	Université de skikda
Invité	Mohamed TOUAHRIA		Université de Sétif -1-

REMERCIEMENTS

Tout d'abord et avant tout, Je remercie Allah le tout puissant et miséricordieux, qui m'a donné la force et la patience d'accomplir ce modeste travail.
Je remercie les membres du département d'Informatique, enseignants et administratifs, qui m'ont aidé tout au long de mon périple à l'université.

J'atteste une reconnaissance indéfectible envers mon encadreur de thèse Mr Mohamed TOUAHRIA, Professeur à l'Université Ferhat Abbas- Sétif -1, pour m'avoir proposé ce sujet et pour son aide, ses encouragements, ses observations avisées, ses conseils éclairés et son support moral, qui m'ont guidé dans les chemins difficiles de la connaissance vers la réalisation de cette thèse qui, je souhaite sera à la hauteur de vos espérances.

Je remercie les membres de jury d'avoir accepté de juger cette thèse.

Un remerciement spécial au Dr. Nouioua Farid pour les nombreuses discussions fructueuses que j'ai pu avoir avec lui et qui m'ont beaucoup aidé à réaliser ce modeste travail.

Je tiens à remercie mes chers collègues : Foudil BELHAJ, Nour elhouda FARES et Ramla BELALTA pour leurs remarques et suggestions très pertinentes sur les premières versions du manuscrit de cette thèse.

Je tiens à remercier tous ceux qui m'ont aidé, de près ou de loin, à mener à bien la réalisation de ce travail.

DÉDICACES

Je dédie ce modeste travail :

- *A la mémoire de ma mère, paix à son âme*
- *A mon cher père.*
- *A mes chers frères et sœurs.*
- *A ma petite famille, ma femme et mes chers enfants.*
- *A tous mes amis et à toute personne qui m'a aidé à réaliser ce travail.*

Résumé

Le modèle de sac de mots est couramment utilisé dans la catégorisation de textes. Le problème principal de ce dernier réside dans le grand nombre d'attributs extraits, cela influe négativement sur les performances des tâches de catégorisation. Pour résoudre ce problème, une méthode de sélection des fonctionnalités est nécessaire. La sélection des fonctionnalités est bénéfique pour réduire la dimensionnalité du problème, elle conduit à minimiser le temps de calcul et à améliorer les performances de la tâche de catégorisation. Dans cette thèse, nous proposons un nouvel algorithme amélioré de l'algorithme original de recherche Sinus Cosinus (SCA) pour la sélection des fonctionnalités, qui permet une meilleure exploration dans l'espace de recherche. Contrairement au SCA qui se concentre uniquement sur la meilleure solution pour générer une nouvelle solution, le nouvel algorithme (ISCA) de notre proposition prend en compte deux positions de la solution : (i) la position de la meilleure solution trouvée jusqu'à présent, et (ii) une position aléatoire appartenant à l'espace de recherche. Cette combinaison nous permet de proposer un algorithme simple capable d'éviter une convergence prématurée et d'obtenir des performances très satisfaisantes. Pour valider le nouvel algorithme ISCA, nous avons effectué une série d'expériences sur neuf collections de textes, où nous avons comparé les résultats expérimentaux avec plusieurs algorithmes de recherche, y compris l'algorithme SCA d'origine et certaines de ses versions améliorées, ainsi que l'algorithme d'optimisation Moth-Flam (MFO). De plus, de l'état de l'art, les algorithmes génétiques (GA) ainsi que les colonies de fourmis (ACO) sont choisis dans notre étude comparative. Nos résultats d'évaluation démontrent la haute performance de notre algorithme ISCA proposé qui le rend très utile pour les problèmes de catégorisation de textes.

Mots clés : Catégorisation des textes, gain d'informations, sélection de sous-ensembles de fonctionnalités, méthodes enveloppes, version améliorée de l'algorithme Sinus Cosinus.

Abstract

Bag of words model is commonly used for text categorization. The main problem of this model lies in the large number of involved features, which influences the categorization task performance. To deal with this problem, feature selection method is necessary. Feature selection is beneficial for reducing the dimensionality of the problem, it leads to minimize the computational time and improve the performance of the categorization task. In this thesis, we propose a new improved algorithm of the original Sine Cosine Algorithm (SCA) for feature selection, which allows for better exploration in the search space. Unlike the SCA which focuses only on the best solution to generate a new solution, the new algorithm (ISCA) of our proposal takes into account two positions of the solution. (i), The position of the best solution found so far, and (ii), a given random position from the search space. This combination allows us to propose a simple algorithm which is able to avoid premature convergence and obtain very satisfactory performance. To validate the new ISCA algorithm, we carried out a series of experiments on nine text collection, where, we compared the experimental results with several search algorithms including the original SCA algorithm and some of its improved versions as well as the Moth-Flam Optimizer (MFO) algorithm. Moreover, from the state of the art, the Genetic Algorithm (GA) and the Ant Colony Optimization (ACO) are chosen in our comparative study. Our evaluation results demonstrate the high performance of our proposed ISCA algorithm which makes it very useful for text categorization problem.

Key words: Text categorization, Information gain, Feature subset selection, Wrapper methods, improved sine cosine algorithm.

الملخص

يستخدم نموذج "كيس الكلمات" غالبا في تصنيف النصوص. المشكلة الأساسية في هذا النموذج تكمن في العدد الكبير من الميزات (الالفاظ) التي يتم استخراجها والتي بدورها تؤثر سلبا على أداء مهمة التصنيف .للتعامل مع هذه المشكلة، يجب اللجوء الى استخدام أسلوب تحديد الميزات .يعتبر تحديد الميزة مفيدًا لتقليل من مشكلة الابعاد المتعددة، مما يؤدي إلى تقليل الوقت الحسابي وتحسين أداء مهمة التصنيف .في هذه الاطروحة، نقترح خوارزمية جديدة محسنة للخوارزمية الأصلية جيب تجب التمام (SCA) لتحديد الميزات النصية، والتي تسمح باستكشاف أفضل في مساحة البحث .على عكس (SCA) الذي يركز فقط على أفضل حل لإنشاء حل جديد، فإن الخوارزمية الجديدة (ISCA) و التي هي من اقتراحنا، تأخذ في الاعتبار موضعين اثنين من الحلول. '1' وضع أفضل حل موجود حتى الان، و '2' موضع عشوائي ينتمي الى حيز البحث .تسمح لنا هذه الاستراتيجية باقتراح خوارزمية بسيطة قادرة على تجنب التقارب السابق لأوانه والحصول على أداء مرضٍ للغاية .للتحقق من صحة خوارزمية ISCA الجديدة، قمنا بإجراء سلسلة من التجارب على مجموعة تسعة قواعد نصية. حيث قمنا بمقارنة النتائج التجريبية بالعديد من خوارزميات البحث بما في ذلك الخوارزمية الأصلية SCA وبعض إصداراتها المحسنة، وكذلك مع خوارزمية فراشة اللهب المحسنة (MFO). علاوة على ذلك، تم اختيار الخوارزمية الجينية (GA) و نسخة محسنة من خوارزمية مستعمرة النمل (ACO) . توضح النتائج التجريبية، الأداء العالي لخوارزميتنا المقترحة (ISCA) مما يجعلها مفيدة جدًا في استخلاص الالفاظ المناسبة لحل مشكلة تصنيف النصوص.

الكلمات الدالة: تصنيف النصوص، كسب المعلومات، اختيار مجموعة فرعية للميزات ، طرق الالتفاف ، خوارزمية البحث جيب التمام المحسنة.

Table des matières

Introduction Générale 0

1. chapitre 01: classification automatique de textes

1.1. Introduction 6

1.2. Classification de textes 6

1.3. Processus de classification de textes 6

1.4. Application de la classification de textes 8

1.5. Problèmes lié à la classification de textes 9

1.6. Les algorithmes de classification 11

1.6.1. Algorithme de Rocchio 11

1.6.2. Naive Bayes 12

1.6.3. Les Machines à Vecteur Support (SVM) 13

1.6.4. Les arbres de décision 14

1.6.5. L'algorithme K-plus proches voisins 15

1.6.6. Les Réseaux de neurones 16

1.6.7. La Régression Logistique 17

1.6.8. Bagging 18

1.6.9. Boosting 19

1.7. Les collections de textes 20

1.7.1. Reuters 21-578 20

1.7.2. 20 new groups 21

1.7.3. Ohsumed 22

1.8. Conclusion 23

2. chapitre 02: indexation des documents textuels

2.1 Introduction 25

2.2 Prétraitement 25

2.2.1 Suppression des caractères inutiles 26

2.2.2 Élimination des mots vides (stop words) 27

2.2.3 Traitement des lettres majuscules 28

2.2.4 La désuffixation (stemming) 29

2.2.5 La lemmatisation 30

2.3 La Représentation de textes 31

2.3.1 Représentation des textes par sac de mots 31

2.3.2 Représentation en phrases 33

2.3.3 Représentation par les N-grammes 33

2.4 La pondération 34

2.4.1 Pondération booléenne 35

2.4.2 Pondération par fréquence de mot 35

2.4.3 Pondération TF-IDF 35

2.5 Réduction de la dimensionnalité 36

2.5.1 Extraction de caractéristiques 37

2.5.1.1 Analyse des composants principaux (PCA) 37

2.5.1.2 Analyse sémantique latente (LSA) 37

2.5.2 Sélection de caractéristiques 38

2.5.2.1 Le choix d'attributs 38

2.5.2.2 Filter 39

2.5.3 Wrapper 43

2.5.4 Embedded 44

2.6 Conclusion 44

3 chapitre 03: Les algorithmes de recherche et la sélection d'attributs.
3.1 Introduction 47
3.2 Les méthodes de sélection 47
3.2.1 Etat de l'art 47
3.3 Les algorithmes de recherche 50
3.3.1 Les stratégies de recherche 51
3.3.2 SFS et SBS 52
3.3.3 La méthode Tabou 53
3.3.4 Recuit simulé 54
3.3.5 Les Algorithmes génétiques (GA) 55
3.3.6 Les colonies de fourmis (ANT) 56
3.3.7 L'optimisation par essaims particulaires (PSO) 57
3.3.8 L'optimisation des essaims particulaires géométrique (GPSO) 58
3.3.9 Algorithme de papillons de nuit (MFO). 60
3.3.10 L'algorithme de Levy Flight (Levy_SCA) 61
3.3.11 L'algorithme SCA pondéré (Weighted SCA) 63
3.4 Conclusion 64

4 Chapitre 04 un nouvel algorithme pour la sélection des termes 66
4.1 Introduction 67
4.2 Notre processus de classification de textes 67
4.3 Représentation des sous-ensembles d'attributs 68
4.4 Evaluation du sous-ensemble de caractéristiques 69
4.5 Un nouvel algorithme de recherche ISCA 70
4.5.1 Introduction 70
4.5.2 Présentation et critique de l'algorithme SCA 70
4.5.3 L'algorithme (SCA) 71

4.5.4 Le nouvel algorithme de recherche ISCA 73
4.6 Conclusion 78
5 Chapitre 5 réalisations et validation 79
5.1 Introduction 80
5.2 Collections de Textes 80
5.2.1.1 Collection Re0 80
5.2.1.2 Collection La1s 80
5.2.1.3 Collection La2s 81
5.2.1.4 Collection Oh0 82
5.2.1.5 Collection Oh5 82
5.2.1.6 Oh10 collection 83
5.2.1.7 Collection Oh15 84
5.2.1.8 Collection FBIS 84
5.2.1.9 Collection Tr41 : 84
5.3 Méthodologie d'évaluation 85
5.4 Résultats et discussion 87
5.4.1 Résultats et Comparaison interne 87
5.4.2 Comparaison Externe 92
5.4.3 Comparaison avec les méthodes filtres 96
5.4.4 Le test statistique de *Wilcoxon* 97
5.5 Conclusion 98
6 Conclusion Générale 101

Table des figures

Figure 1-1 Processus de catégorisation automatique des textes 7
Figure 1-2 classification liniare par SVM 13
Figure 1-3 Arbre de décision appliqué sur la Base Reuters avec l'outil Weka. 14
Figure 1-4 Illustration d'une classification par l'algorithme K-nn, cas de deux classe A et B. 16
Figure 1-5 un schéma du perceptron 17
Figure 1-6 : schema general de Bagging 19
Figure 2-1 Exemple de suppression des caractères inutiles 27
Figure 2-2 L'élimination des mots vides 28
Figure 2-3 La suppression des numéros 28
Figure 2-4 Le traitement des majuscules 29
Figure 2-5 Exemple d'application de la racinisation 30
Figure 2-6 La représentation par sac de mots, exemple de trois documents 32
Figure 2-7 processus de sélection d'attributs 38
Figure 2-8 schéma de l'approche 'Filter' 40
Figure 1-9 : schéma général d'un Wrapper 44
Figure 3-1 classification des algorithmes de recherche 52
Figure 3-2 : pseudo code de l'algorithme Tabou 54
Figure 3-3 : pseudo code de la méthode Recuit simulé 55
Figure 3-4 : Schéma général d'algorithmes génétiques 56
Figure 3-5 L'état initial du système. 57
Figure 3-6 Comportement des Fourmis après au début de simulation. 57
Figure 3-7 Comportement des Fourmis après une période de temps. 57
Figure 3-8 schémas de principe du déplacement d'une particule. 58
Figure 3-9 sélection d'un terme à partir de 3 parents avec le masque 3pBMCX (Karabulut, 2013) 59
Figure 3-10 pseudo code de l'algorithme GPSO 59
Figure 3-11 mouvement spirale des papillons de nuit sur la source de lumiere. 60
Figure 3-12 illustrationde la fonction Levy Flight 62
Figure 3-13 les stratégies de l'intensification et de la diversification de l'algorithme SCA. . 63
Figure 4-1 Notre processus de classification de textes. 68

Figure 4-2 Stratégie adoptée pour la classification de textes. 68
Figure 4-3 l'évolution de la fonction Sinus suivant le paramètre r1, (a=2). 74
Figure 4-4 exploration et exploitation des algorithmes SCA et ISCA, l'algorithme SCA en haut;et l'algorithme Sin Cosine amélioré en bas75Figure 5-1 Moyennes de Micro-F1 de SCA, OBL-SCA, Levy_SCA , Weighted_SCA , MFO, ACO, GA et de l'ISCA proposé pour la sélection d'attributs sur les 9 collections en augmentant le nombre de termes. 91
Figure 5-2 Moyennes de Micro F1 de SCA, OBL-SCA, Levy_SCA , Weighted_SCA , MFO, ACO, GA et de l'ISCA proposé pour la sélection d'attributs sur les 9 collections en augmentant le nombre de termes. 91
5Figure 5-3 Moyennes de Micro F1 de SCA, OBL-SCA, Levy_SCA , Weighted_SCA , MFO, ACO, GA et de l'ISCA proposé pour la sélection d'attributs sur les 9 collections en augmentant le nombre de termes. 95
Figure 5-4 Moyennes de MacroF1 de SCA, OBL-SCA, Levy_SCA , Weighted_SCA , MFO, ACO, GA et d'ISCA proposé pour la sélection d'attributs sur les 9 collections en augmentant le nombre de termes. 96

Liste des Tableaux

Tableau 1-1 nombre de documents d'entraînements et de testes par catégorie de la base Reuters 21578 ModApte Split. 21
Tableau 1-2 nombre de documents d'entraînements et de tests par catégorie de la base 20-Newgroups. 22
Tableau 1-3 distribution des documents par catégorie dans la collection Ohusmed. 23
Tableau 5-1 Nombre de documents par classe pour la collection Re0. 81
Tableau 5-2 Nombre de documents par classe de la collection La1s. 81
Tableau 5-3 Nombre de documents par classe de la collection La2s. 82
Tableau 5-4 Nombre de documents par classe de la collection Oh0. 82
Tableau 5-5 Nombre de documents par classe de la collection Oh5. 83
Tableau 5-6 Nombre de documents par classe de la collection Oh10. 83
Tableau 5-7 Nombre de documents par classe de la collection Oh15. 84
Tableau 5-8 Nombre de documents par classe pour la collection FBIS. 85
Tableau 5-9 Nombre de documents par classe pour la collection Trec41. 85
Tableau 5-10 les paramètres de SCA, OBL-SCA, Levy_SCA , Weighted_SCA , et de l'algorithme ISCA proposé. 88
Tableau 5-11 Moyennes de précision (P), de rappel (R) et de F-mesure (F1) de SCA, OBL-SCA, Levy_SCA et l'ISCA proposé les 9 collections de texte 88
Tableau 5-12 Micro-F1 et Macro-F1 de SCA, OBL-SCA, Levy_SCA , Weighted_SCA , MFO, ACO, GA et de l'ISCA proposé sur les 9 collections de textes (en pourcentage) 89
Tableau 5-13 SCA, OBL-SCA, Levy_SCA , Weighted_SCA , MFO, ACO , GA et les paramètres de paramètres ISCA proposés 92
Tableau 5-14 mo yen s de précision (P), le rappel (R) et F-mesure (F1) de SCA, OBL-SCA, Levy_SCA et l'ISCA proposé por les 9 collections. 93
Tableau 5-15 Micro-F1 et Macro-F1 de SCA, OBL-SCA, Levy_SCA , Weighted_SCA , MFO, ACO, GA et de l'ISCA proposé sur les 9 collections de textes (en pourcentage) 94
Tableau 5-16 Evaluation de performances de l'algorithme ISCA contre les filtres IG, Corrélation et Importance, sur les 9 collections de texte. 96
Tableau 5-17 **les valeur de p (** *p* –Values) du Wilcoxon test (ranksum) sur toutes les analyses (les *p*> = 0,05 sont soulignés). 98

Liste d'abréviations

TC	Text Categorization
FS	Feature selection
IG	Information Gain
GA	Genetic algorithms
LSA	Latent semantic analysis
PCA	principal component analysis
TF	Term Frequency
IDF	Inverse Document Frequency
TF-IDF	Term Frequency- Inverse Document Frequency
PSO	Particle swarm optimization
GPSO	Geometric Particle Swarm Optimization
SVM	Support Vector Machines
SFS	Sequential Forward Selection
SBS	Sequential Backward Selection
TS	Tabou Search
SCA	Sine Cosine Algorithm
ACO	Ant Colony Optimisation
MFO	Moth-Flame optimization
ISCA	Improved Sine Cosine Algorithm

Introduction Générale

Introduction Générale

- ✓ ***Contexte de travail***

De nos jours, le réseau internet est en train de s'enrichir continuellement par de nouveaux contenus. La production des documents électroniques est en croissance continue. L'utilisation du courrier électronique est devenue une habitude journalière. Des milliers si ce n'est pas des millions de personnes ou d'entreprises communiquent régulièrement. Par ailleurs, les réseaux sociaux, à savoir Facebook, twitter, etc. transmettent eux aussi et stockent des millions de SMS (Short Messages String) par jour. Toute cette immense quantité de données électroniques produites sans compter d'autres types de ressources comme les sites web d'entreprises ou d'individus, les blogs, et ainsi des documents autrefois manuscrits sont aujourd'hui disponibles sous format numérique. A cet effet, on devrait disposer de techniques à la fois automatiques et efficaces face à cette importante quantité de textes électroniques. En conséquence, on aura les avantages suivants : l'accès instantané à l'information, l'organisation et exploitation des documents, et même l'extraction des connaissances à la fois utiles et qui sont cachées au préalable. L'intelligence artificielle (IA), et plus en particulier l'apprentissage automatique s'avère, potentiellement, capable de nous répondre à tous ces besoins. L'apprentissage automatique est un processus d'induction général qui permet la construction automatique de classificateurs (McCallum & Nigam, 1998). Plusieurs disciplines ont vu le jour dans cette dernière décennie, notamment, la recherche d'information, l'indexation automatique, la catégorisation et le regroupement parmi d'autres.

Un de ces domaines prometteurs est la catégorisation automatique de textes. Imaginons une base de données textuelle considérable (corpus de textes), l'accès aux documents serait plus aisé s'ils étaient répartis dans des dossiers (un ensemble de catégories) en fonction de leur sujet. Certainement, un expert humain, un linguiste, peut nous lire tous les textes et les classer manuellement. Mais la tâche s'avère colossale s'il fait face à des centaines, voire des milliers de documents. En revanche, Il est nécessaire de faire appel à une application informatique qui assignera automatiquement ces textes à un ensemble prédéfini de catégories selon leurs contenus. C'est également l'une des définitions de la classification automatique de textes. Cette tâche a plusieurs applications, notamment l'identification de la langue, la reconnaissance d'écrivains, la catégorisation de documents multimédia, et bien d'autres. Ainsi,

elle peut servir d'autres applications parmi lesquelles le filtrage dans le cas de la détection de spams (les courriers indésirables), ainsi que le filtrage comme par exemple dans la diffusion sélective d'information.

- ✓ ***Position du problème***

Classer des textes automatiquement nécessite d'abord une phase de représentation automatique de textes, aussi appelé une étape de numérisation ou l'indexation. L'un des modèles les plus connus est le modèle sac de mots, (bag of words en anglais). Malheureusement, le nombre excessif des termes générés par ce dernier pose un grand problème : on se retrouve face à des centaines de milliers de termes, en général. Cette taille impressionnante du vocabulaire peut s'avérer un obstacle à l'utilisation des algorithmes de catégorisation. Par conséquent, il est souvent utile, et parfois nécessaire, de réduire celle-ci à une taille plus compatible avec les méthodes de résolution, malgré si cette dernière peut conduire à une légère perte d'informations. C'est pour cette raison que certaines techniques ont été mises en place pour réduire la dimension du vocabulaire. En effet, il existe deux grandes familles. La sélection d'attributs (*''feature selection''*), prend les attributs d'origine et conserve seulement ceux jugés utiles pour la prochaine étape qui est la classification. Une fonction d'évaluation est assignée au processus de sélection. Si le critère d'évaluation est de même type que celui de la phase de classification, ce type de sélection est appelé ''Filtre''. Dans le cas contraire, il s'agit de ''Wrapper''. La deuxième catégorie des méthodes de sélection d'attributs, effectue plutôt une extraction d'attributs (''*feature extraction''*) à partir des attributs originaux. Carrément, de nouveaux attributs seront générés en faisant soit des regroupements ou des transformations.

L'analyse des techniques de sélection existantes (Filtre, Wrapper) nous a permis de mettre en évidence un certain nombre de faiblesses. On note la complexité très élevée des approches "Wrapper" ce qui conduit à un temps d'exécution considérable, et au même temps elles sont couteuses en termes de ressources. Dans certaines situations il est impossible d'appliquer ces méthodes vu la longueur du vecteur de caractéristiques générés lors de la phase d'indexation. Contrairement aux techniques Wrapper, les méthodes Filtres montrent une très bonne efficacité face au grand nombre de caractéristiques . En revanche, les Filtres présentent une limitation vue la redondance des attributs sélectionnés et l'information partagée entre attributs sélectionnés. En conséquence, ces derniers défauts influent négativement sur la performance des classificateurs utilisés dans la phase de classification.

✓ *Contribution*

Dans cette thèse, on compte résoudre le problème de la haute dimensionnalité, et en particulier sur un cas de bases de données documentaires. Vu les limites et les inconvénients des techniques cités précédemment, nous proposons une nouvelle stratégie de sélection de caractéristiques qui est très efficace face à ce problème. Notre première contribution consistera tout d'abord à effectuer deux phases de sélection de termes : dans la première on effectue un filtrage rapide des caractéristiques en utilisant la mesure statistique (le Gain d'information (GI)) qui permet d'évaluer les caractéristiques individuellement (*feature ranking*). Ensuite, une deuxième étape qui est une méthode de type Wrapper. Celle-ci permet d'extraire un sous-ensemble optimal de termes à partir des termes produits par la première étape Filtre. Autrement dit, les termes résultant de la première étape seront des inputs pour la seconde étape. Le sous ensemble final de termes est obtenu en se basant sur un algorithme de recherche et un critère d'évaluation. A ce point-là, il est temps de dévoiler notre deuxième contribution qui est la proposition d'un nouvel algorithme de recherche. Cet algorithme servira à la sélection des termes à partir d'une collection de textes. Sine Cosine Algorithm (SCA) est l'un des puissants algorithmes de recherche, proposé récemment par Mirjalili, (Mirjalili, 2016). Cet algorithme est, d'origine, conçu pour la résolution des problèmes d'optimisation globale. La réussite impressionnante de cet algorithme et son avance présentée dans les expériences. En outre, plusieurs travaux de recherche se basant sur l'algorithme SCA ont été proposés dans la résolution des problèmes d'optimisation. Cela met en valeur la crédibilité et l'efficacité de cet algorithme. La qualité remarquable de SCA nous a ainsi motivé pour la proposition d'un nouvel algorithme appelé Improved Sine Cosine Algorithm (ISCA). Ce nouvel algorithme ISCA sera donc une version amélioré de l'algorithme SCA original et sera particulièrement conçu et validé pour la sélection des termes à partir de bases de données textuelles.

Pour valider et évaluer notre nouvel algorithme ISCA, nous allons utiliser dans l'expérimentation 09 Banc d'essais. Elles étaient construites à partir de trois Benchmarks textuelles très reconnues, notamment Reuters21-578, TREC et Ohsumed. Dans la phase de classification, le classificateur Naive-Baiyse est utilisé. En plus, une étude comparative sera accompagnée dans cette expérimentation dans laquelle notre algorithme ISCA sera testé contre sept(7) algorithmes de recherche très connus, particulièrement : les algorithmes génétiques, les colonies de fourmis et bien sûr avec l'algorithme original SCA. On outre, et

pour avoir plus de crédibilité, une seconde expérimentation qui consiste à valider et évaluer notre approche face à d'autres méthodes de nature purement filtre sera délivrée dans la partie d'étude expérimentale de cette présente thèse.

- ✓ ***Organisation de la thèse***

La présente thèse se structure comme suit :

Le chapitre 1, présente un aperçu sur le processus de la classification de textes. Les algorithmes de classifications, les problèmes liés et ainsi que les domaines d'applications dans un contexte de catégorisation de textes sont décrits dans ce chapitre. Il contient aussi une explication détaillée sur les collections de textes les plus utilisé dans la catégorisation automatique.

Le chapitre **2**, est consacré à la phase de prétraitement de textes. Il contient à la fois le processus de nettoyage, les méthodes de représentation de textes et ainsi que les mesures de pondération de termes. Il présente également un aperçu sur les approches et les méthodes de réduction de la dimensionnalité.

Dans le chapitre **3**, nous allons présenter un état de l'art sur les méthodes de sélection d'attributs. On va introduire les différentes stratégies de recherche. Des algorithmes de sélection des attributs seront décrits et présentés en détail, à savoir Les concepts, les caractéristiques et également les avantages et l'inconvénient de chacun deux.

Le chapitre **4,** expose le nouvel l'algorithme de recherche ISCA qui est proposé pour la sélection de termes dans un contexte de catégorisation de tests. La stratégie de diversification de cet algorithme (ISCA) est bien décrite et également la nouvelle formulation des équations de génération de nouvelles solutions. En plus, une présentation est donnée sur le processus de classification dans laquelle on mit l'accent sur partie de de sélection d'attribut, la présentation de sous-ensemble d'attributs, ainsi que leur évaluation adoptée dans cet algorithme ISCA.

Le Chapitre **5**, est consacré à la partie réalisation de ce mémoire. Notre algorithme de recherche ISCA, sera évalué sur un grand nombre d'expériences. Afin de valider notre algorithme ISCA nous allons faire une comparaison avec d'autres algorithmes de recherche connus dans le domaine de sélection des caractéristiques (FS) aussi bien que ceux déjà appliqués dans un problème de TC, à savoir les algorithmes génétiques (GAs), les colonies de fourmis (ACO) et également l'algorithme original Sine Cosine Algorithm (SCA). A noter que la plupart des résultats de performance communiqués sont indiqués à la fois qualitativement et quantitativement.

Enfin, la conclusion générale présente la contribution apportée ainsi que les pistes définissant des perspectives possibles pour de futurs travaux.

Chapitre 01

Classification automatique de textes

1.1. Introduction

Dans ces dernières décennies, il y a eu une croissance importante du nombre de documents électroniques textuels. De plus, l'accès à l'information est devenu un besoin mais aussi un problème de vie au quotidien. Dans ce contexte, se requérir à des méthodes d'apprentissage automatique est devenu un besoin primordial. Les méthodes d'apprentissage automatique sont capables de classer les textes de façon automatique et fiable dont lesquelles de nombreuses approches d'apprentissage automatique ont dépassé celle du traitement du langage naturel. Le succès de ces algorithmes d'apprentissage repose sur leur capacité de générer des modèles complexes et d'extraire à partir de données des connaissances qui sont non seulement cachées , mais de plus qui sont pertinentes et utiles. Cependant, développer des structures et des techniques adéquates pour la classification de textes était, et il est toujours, un défi pour les chercheurs. Dans ce chapitre, nous présentons un aperçu sur les algorithmes de classification des textes. Nous décrivons les algorithmes et les techniques de catégorisation de textes existants, ainsi que leurs méthodes d'évaluation.

1.2. Classification de textes

La Classification de textes est le processus permettant d'associer un nouveau texte libre à une catégorie (classe), en fonction des informations qu'il contient. Dans cette optique, il faut préparer un ensemble de textes préalablement étiquetés, dit ensemble d'apprentissage, à partir duquel nous générons un modèle de prédiction le plus adéquat, le modèle le plus performant.

Formellement, le problème TC est défini comme une tâche d'assigner une valeur booléenne (T, F) à la paire (di, ci) $\in$ D x C, (T : document appartient à la catégorie), (F : document n'appartient pas à la catégorie). Dans cette notation, D = {d1,. . . ,dn} représente un ensemble de documents à catégoriser et C = {c1,. . . ,cm} correspond aux catégories de texte prédéfinies.

1.3. Processus de classification de textes

Un processus de catégorisation des documents textuels, en général, peut se composer des quatre phases suivantes : Extraction de caractéristiques, réductions de dimensionnalité, classification et évaluations des performances. La figure1.1 ci-dessous expose les quatre

phases du système dans leur ordre d'exécution. L'entrée initiale du système possède un certain ensemble de données, de textes bruts, ensemble de documents, appelé aussi jeux de données (Benchmarks).

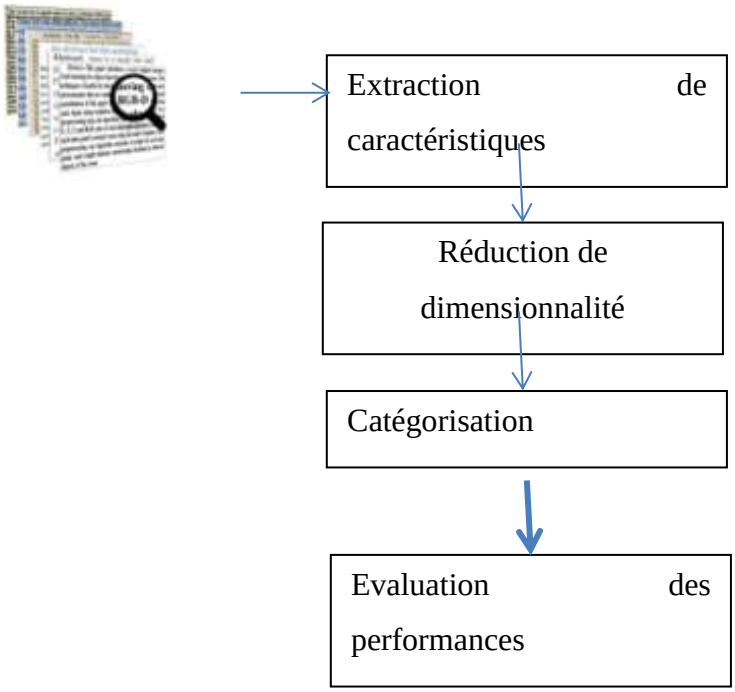


Figure 1-1 Processus de catégorisation automatique des textes

En général, les ensembles de données textuelles contiennent des unités de textes, documents. Un document se compose d'un nombre de phrases de sorte que chaque phrase comprend une suite de mots sépares évidement par des espaces. De leurs côtés, les mots, sont présentés par une chaine de caractères. Dans la classification supervisée, i.e., catégorisation automatique, chaque document est étiqueté avec une valeur qui est la classe. Cette étiquette est souvent désignée par un expert. Un système de catégorisation automatique devrait tout d'abord, être capable, d'acquérir des données textuelles dans différents formats, extension, tel que : html, xml, pdf ou doc…etc. Une fois ces données sont chargées, un traitement primordial qui est la numérisation, prétraitement (Feature extractor), aussi appelé segmentation de textes (Tokenization), est appliqué dont le but est d'obtenir une représentation de textes. Généralement on obtient un vecteur de caractéristiques sur lequel on peut appliquer les algorithmes de classification. Le modèle Sac de Mot (Bag of Words) (BoW) est le modèle le plus connu et le plus simple dans la représentation de textes. L'inconvénient majeur de ce dernier réside dans la longueur de son vecteur de caractéristiques. Ensuite, le processus de TC

est enchainé au choix par une deuxième étape qui est la réduction de la dimensionnalité. Cependant, cette dernière est parfois indispensable suivant la dimensionnalité traitée. En outre, cette étape permet non seulement de réduire la taille du vocabulaire, mais aussi elle conduit à des bonnes performances des classificateurs. La phase de catégorisation est la plus essentielle parmi toutes les phases indiquées dans ce processus. Elle comporte deux étapes, une première qui est l'entrainement, ou la génération du modèle, et une deuxième étape qui est l'étape de prédiction. Les données de leurs côté sont subdivisées aussi en deux parties. Une partie ou un ensemble pour la phase d'entrainement et une autre partie de données qui est un ensemble de test consacré à la phase de perdition. Plusieurs algorithmes d'apprentissage sont appliqués à la CT tel que les Support Vecteurs Machines (SVM), les arbres de décision et l'algorithme Naive Bayes. Ce dernier est très reconnu dans plusieurs applications de CT. Finalement, La dernière partie du processus CT est l'évaluation. Connaître la performance est essentielle à l'utilisation des algorithmes de classification. Cela permet d'en choisir le meilleur, pour un cas d'application, selon les résultats de performances obtenus. À noter que la performance d'un algorithme donné varie d'un domaine à un autre. Par ailleurs, Il y a de nombreuses métriques d'évaluation de performances qui servent à évaluer les algorithmes de classification. Notamment, le Rappel et la Précision qui sont les méthodes d'évaluation les plus simples.

1.4. Application de la classification de textes

La catégorisation de textes a été utilisée pour un nombre d'applications différentes. Quelques exemples de domaines dans lesquels la classification textuelle est souvent utilisée sont les suivants:

- **Filtrage et organisation des nouvelles:**

La plupart des services d'information sont aujourd'hui de nature électronique dans laquelle un grand nombre d'articles de journaux sont créés journalièrement par les presses. De ce fait, il est difficile de les organiser manuellement. La classification automatique peut être très utile pour la catégorisation des divers portails web (Lang, 1995a), (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994). Cette application est aussi appelée filtrage de textes.

- **Organisation et extraction des documents :**

L'organisation et l'extraction des documents est, généralement utile pour de nombreuses applications notamment, le filtrage et l'organisation des articles d'information. On trouve plusieurs méthodes supervisées qui peuvent être utilisées pour l'organisation de documents, et cela dans de nombreux domaines à savoir, la classification hiérarchique des collections de documents. A noter que, les organisées de manière hiérarchique peut être particulièrement utile pour la navigation et également pour la récupération (Chakrabarti, Dom, Agrawal, & Raghavan, 1997).

- **La fouille d'opinion :**

Les opinions des clients sont généralement exprimées en petits textes. Ces derniers peuvent nous servir à la conduite des informations utiles dans une revue. Un tel exemple de classification utilisé pour effectuer l'extraction d'opinion est discuté dans (Bakshi, Kaur, Kaur, & Kaur, 2016), (L. Zhang & Liu, 2017).

- **Classification des Emails et filtrage des Spams:**

La classification des Emails et filtrage des Spams, appelé aussi filtrage des emails, (Carvalho & Cohen, 2005). Classer automatiquement les courriels est souvent judicieux afin de déterminer que ce soit le sujet, ainsi que les courriels électroniques indésirables, (Cohen, 1996).

1.5. Problèmes lié à la classification de textes

En plus des problèmes propres à la discipline de l'apprentissage automatique, notamment la classification, à savoir le sur-apprentissage ou la subjectivité de l'expert et/ou le déséquilibrage, etc. il existe également d'autres types de problèmes induits cette fois-ci par la nature des données. Pour le traitement de textes, on peut citer par exemple ceux de : synonymes, ambiguïté et mot-composé …etc.

Dans la suite, nous présentons certains problèmes que nous rencontrons lors de la classification automatique de textes.

- **La haute dimensionnalité :**

  Le modèle sac de mots génère souvent des centaines de milliers de termes. Ceci affecte la qualité de la classification, ce qui entraîne non seulement le coût en temps des algorithmes de classification, mais conduit souvent à ce qu'on appelle un sur-apprentissage. De plus, notre vecteur de mots dont la plupart des variables sont vides est un vecteur creux, ce qui pose un problème secondaire à ceux de la dimensionnalité et du sur-apprentissage.

- **Complexité de l'algorithme :**

  Dans la catégorisation de textes, la grande taille du vecteur de mots ainsi que le nombre d'observations affectent l'efficacité et les performances des algorithmes d'apprentissage.

- **Sur-apprentissage :**

  Le nombre important de termes générés par la phase de prétraitement va produire le phénomène de sur-apprentissage pour les algorithmes de classification. Cela veut dire que l'algorithme classe bien les exemples dans la phase d'apprentissage, mais, il a du mal à les classer pour la phase de test.

- **Déséquilibrage :**

  Dans la plupart des benchmarks, le nombre d'observations par classe est souvent déséquilibré, inéquitable, et pour certaines classes le nombre d'exemples positifs est faible comparé à celui des instances négatives. Ceci va créer une difficulté supplémentaire au classement dans lequel les classes majoritaires seront bien classées et les autres seront mal classées.

- **Ambiguïté (polysémie) :**

  Dans le langage naturel, un même mot peut avoir plusieurs sens différents et plusieurs définitions lui sont associées. Par exemple, le terme jaguar peut référer à l'animal

ainsi qu'à la voiture. Si on ne les distingue pas, c.-à-d. on les représente par la même variable (terme), cela va sûrement réduire la qualité de la classification ultérieurement.

- **Synonymie :**

  Les synonymes sont des mots ou des expressions qui indiquent le même sens. La non-prise en considération de ces synonymes dans la phase de représentation des documents va étendre de plus notre vecteur de mots et par conséquent, on aura une faible précision dans les performances.

1.6. Les algorithmes de classification

Il existe plusieurs algorithmes dans l'apprentissage automatique et notamment la classification. Ces derniers sont de différents types mais ayant tous la même intention d'avoir une bonne performance tôt en étant efficace. Chacun d'eux a ces propres avantages et inconvénients. Cependant, et quant à la construction d'algorithmes on trouve généralement deux phases durant le processus d'apprentissage. Une qui est la phase d'entrainement dans laquelle on essaye de générer un modèle ou d'estimer une fonction pour relier une catégorie avec un document. La deuxième permet de décider pour chaque document à quelle catégorie il appartient. Par ailleurs, les algorithmes de classification pourraient avoir plusieurs inspirations ou types, à savoir : mathématiques, probabilistes, ensemblistes, floue, règles de décisions…etc.

Dans les sous-sections suivantes on va présenter quelques algorithmes de classification qui sont appliqués dans la classification de textes ainsi très reconnus dans plusieurs domaines d'application de la classification.

1.6.1. Algorithme de Rocchio

L'algorithme de Rochio est l'un des algorithmes les plus classiques reconnus dans la discipline de recherche d'information RI. Il a été conçu au début et utilisé dans une technique connue sous le nom de la réinjection de pertinence (Relevance feedback). Cette dernière est l'une des techniques de modification de requêtes les plus utiles dans la RI. Cette méthode a pour objectif de modifier la requête d'origine afin d'améliorer le résultat de recherche. Plus tard, l'algorithme de Rochio a été adopté pour la première fois dans la classification textuelle par Hull (Hull, 1994), et en particulier dans le modèle de représentation sac de mots. La

méthode de classification est très simple, un document est classé dans une classe Ci, si la distance entre son vecteur de caractéristiques et le vecteur du document prototype (centroïde) de cette classe est minimal parmi les autres distances d'autres classes. Les poids du vecteur centroïde relatif à une catégorie donnée sont calculés comme c'est exprimé dans la formule 1-1.

$$\boldsymbol{w_{ki} = \beta * \sum_{\{dj \in POSi\}} \frac{w_{kj}}{|POSi|} - \gamma * \sum_{\{dj \in NEGi\}} \frac{w_{kj}}{|NEGi|}} \qquad \textbf{(1-1)}$$

Où w_{kj} est le poids du terme t_k dans le document dj. POSi = {ensemble de documents d'entrainement étiquetés par Ci}, et NEGi = {ensemble de documents d'entrainement étiquetés par $\overline{Ci}$}. Dans cette formule, β et γ sont des paramètres déterminés par l'utilisateur qui permettent d'ajuster l'importance relative des documents d'entrainement positifs et négatifs. Généralement, le poids des documents négatifs est toujours allégé en fixant des grandes valeurs pout β et des petites valeurs pour et γ (Cohen & Singer, 1999).

1.6.2. Naive Bayes

Naïve Bayes (Mccallum et Nigam, 1998) se base sur la règle de Bayes pour prédire qu'un document appartient à une catégorie donnée. Dans cet algorithme la catégorie la plus probable sera choisie. Plus particulièrement, l'algorithme Naïve Bayes utilise la probabilité conditionnelle d'ensemble des termes t et d'une catégorie c pour calculer la probabilité d'un terme qui appartient à un certain nombre de documents dj. A noter que, l'hypothèse bayésienne suppose que tous les termes t d'un document d sont indépendants les uns des autres selon le contexte d'une catégorie c. le calcul de la probabilité qu'un document dj appartient à une catégorie ci est présenté dans Equation 1-2 :

$$p(d_i|c_i) = \prod_{k=1}^{n} \frac{p(w_{jk}|c_i)}{p(c_i)} \qquad (1\text{-}2)$$

Dont, P (di|ci) désigne la probabilité que le document dj appartient à une catégorie ci. La probabilité conditionnelle P (wik|ci) signifie la probabilité du terme tk de document dj qui appartient à une catégorie ci. Enfin, n est le nombre de termes dans le document dj et ci.

1.6.3. Les Machines à Vecteur Support (SVM)

Les séparateurs à vaste marge, connus par nom : les Machines à Vecteur Support (SVMs), ou en anglais Support Vector Machines, ont été développés par Cortes et Vapnik en 1995 (Cortes & Vapnik, 1995). SVMs sont conçues au début pour la classification binaire, les problèmes de discrimination. Contrairement aux autres stratégies de certains algorithmes qui visent à trouver juste une séparation, peu importe la qualité, entre les deux classes lors de la génération de leurs modèles, SVMs non seulement tendent vers trouver une séparation mais aussi elles essayent de générer un séparateur optimal dans un sens de tracer un hyperplan, qui maximise la distance (la marge) entre les bornes ou les frontières des nuages de points positifs et négatifs à la fois.

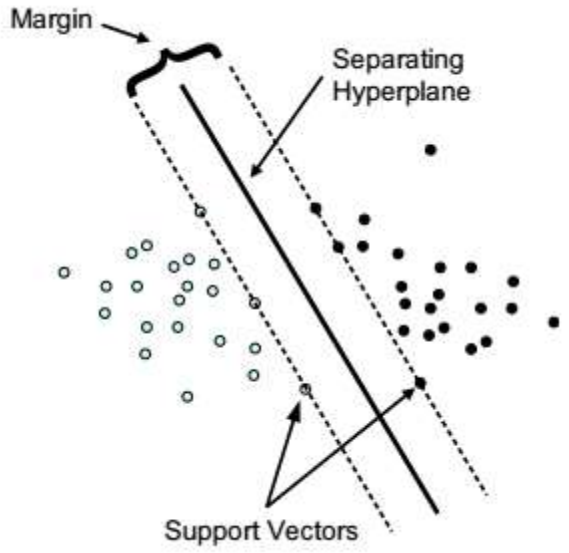


Figure 1-2 classification liniare par SVM

Dans la figure1.2 ci-dessus on montre un cas de classification binaire. Les points d'extrémités des deux classes s'appellent Vecteurs Supports sur lesquelles on construit l'hyperplan qui occupera la position du milieu entre eux. Plusieurs extensions ont été développées par la suite tel que la régression, et la classification multi-classe dans laquelle les SVMs utilisent la technique un-contre-un en ajustant toutes les sous-classes binaires. La décision de classification d'une instance donnée est obtenue par un mécanisme de vote.

Pour les cas non-linéairement séparables les SVMs utilisent la technique des noyaux (Kernels) afin de transformer le problème initial de son espace d'origine vers un autre de plus

grande dimensions. En effet, et avec transformation on se retrouve devant le fait de résoudre un problème simple et qui est linéairement séparable.

1.6.4. Les arbres de décision

Les arbres de décision sont un outil d'apprentissage automatique basé sur la structure d'arbre. L'avantage majeur de ce type est qu'il est très semblable au raisonnement humain, de ce fait il est très facile d'expliquer les résultats obtenus et son comportement. La construction de l'arbre est basée sur un algorithme dans lequel on utilise des mesures statistiques pour faire des séparations (split) entre les données à travers une variable, un attribut, sélectionnée. Particulièrement, dans chaque itération et pour un ensemble de données, on calcule une mesure pour chaque variable, et la meilleure parmi eux sera choisie comme Nœud. Les différentes valeurs figurées dans cette variable seront les branches directes du nœud en cours. Chaque chemin produit par une branche nous ramène vers un nouveau sous-ensemble de données. Le critère d'arrêt est le suivant : si un sous-ensemble donné dans lequel on trouve que toutes les instances appartiennent à la même classe alors on va créer une feuille d'arbre, et la fin du processus pour ce chemin. Pour le cas contraire, un sous-ensemble avec des instances de différentes étiquettes, le processus de construction du nœud est repris. Les algorithmes ID3, C4.5 et CART sont les plus reconnus parmi d'autres. Ci-dessous, la figure1.3 montre un arbre de décision généré par l'algorithme C4.5 sur un jeu de données Reuters 21-578, notamment la classe CORN.

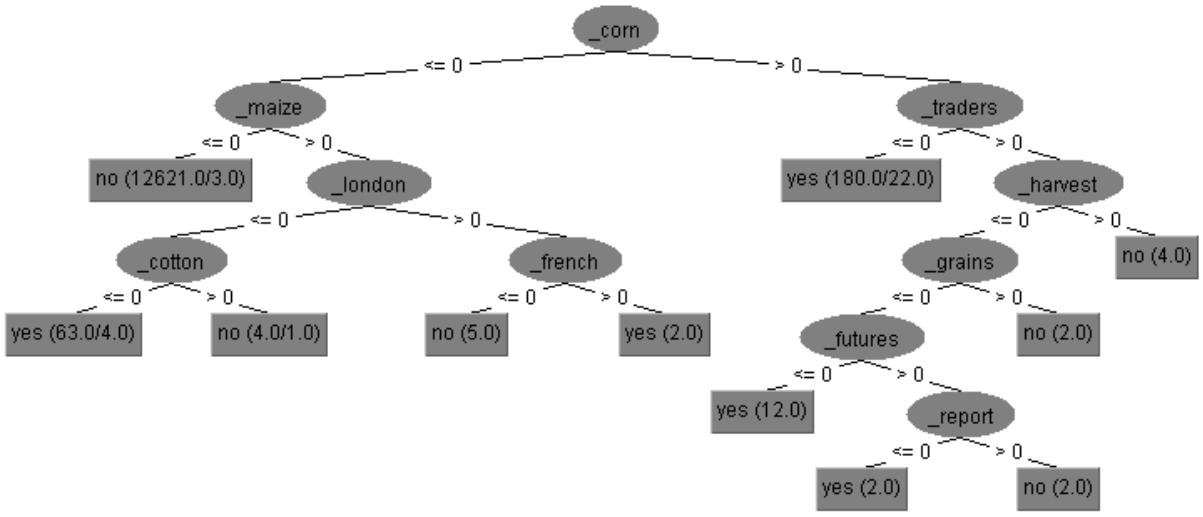


Figure 1-3 Arbre de décision appliqué sur la Base Reuters avec l'outil Weka.

1.6.5. L'algorithme K-plus proches voisins

Ce type d'algorithme appartient aux méthodes d'apprentissage à base de cas (instance-based learning). Il ne comporte pas de phase d'entraînement et les documents d'apprentissages sont juste projetés dans un plan. La décision qu'un document sera assigné à une classe Ci est prise si la majorité des documents de son voisinage sont étiquetés par celle-ci. En d'autres termes, on calcule un score, généralement une distance entre le document en cours de test et les autres documents d'apprentissage. Les documents les plus proches, ayant une distance relativement minimale aux autres, seront exclusivement pris pour prendre la décision. La classe dominante de ce groupe sélectionné, évidemment la classe la plus fréquente, sera choisie et affectée au document en test. En outre, le nombre de documents choisis pour la prédiction est déterminé par un paramètre K attribué par l'utilisateur et qui est compris entre 1 et 20, généralement (Chantar & Corne, 2011). Pour les mesures de Distance, on trouve différentes fonctions. Concernant les formules ci-dessous on considère que X et Y sont deux exemples avec X =(X_1,X_2,…,Xk) et Y=(Y_1,Y_2,….,Yk) respectivement et D(X,Y) est la distance.

Algorithme KNN (DATA, k, distance)
1. Charger les données 2. Initialiser la valeur de k 3. Parcourir et Calculer la distance entre le point teste et l'ensemble des points d'apprentissage 4. Trier dans une liste les documents (points) d'apprentissage en ordre croissant en fonction des valeurs de distance. 5. Obtenir le top k documents à partir de notre liste triée 6. déterminer la classe la plus fréquente de ces documents 7. Retourner la classe trouvée Fin d'algorithme

Euclidien $$D(X,Y) = \sqrt{\sum_{i=1}^{k}(Xi - Yi)^2} \quad (1\text{-}3)$$

Manhattan $$D(X,Y) = \sum_{i=1}^{k}/Xi - Yi/ \quad (1\text{-}4)$$

Minkowski $$D(X,Y) = \sqrt[p]{\sum_{i=1}^{k}/Xi - Yi/^{\mathrm{p}}} \quad (1\text{-}5)$$

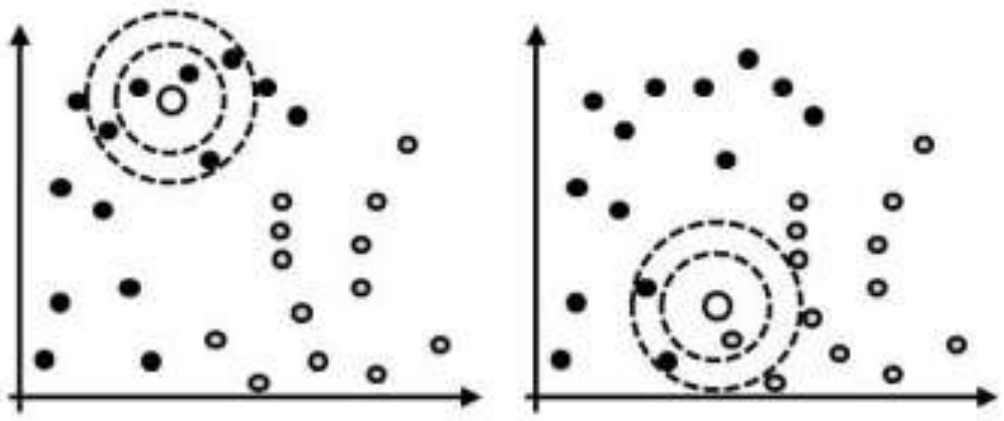


Figure 1-4 Illustration d'une classification par l'algorithme K-nn, cas de deux classe A et B.

Pour déterminer les documents les plus proches, toutes les mesures de similarité ou distance peuvent être utiles. Notamment, la distance Eucleidiene, Manhatan et la similarité Cosinus parmi d'autres. La construction d'un classificateur k-NN peut contenir dans son expérimentation un ensemble de validation afin de déterminer le seuil k qui indique combien de documents d'entrainements sur lesquelles la prise de décision sera établie. Par ailleurs, Les expériences ont montré que l'augmentation de k n'augmente pas forcément la performance du classificateur K-nn , (Uğuz, 2011).

1.6.6. Les Réseaux de neurones

Les Réseaux de Neurones sont inspirés du système neuronal biologique. Le célèbre modèle Perceptron, est le premier modèle développé par Frank Rosenblatt en 1958, (Rosenblatt, 1958). Le réseau Perceptron est une structure mono couche, dont on a plusieurs entrées Xi et une seule sortie Y. Ces entrées sont pondérées par des poids Wi appelé poids synaptiques. La sortie Y, la valeur de classe, est calculée récursivement sur la base d'une fonction prédéfinie et en fonction des entrées Xi, le vecteur de caractéristiques, et leurs poids

synaptiques qui sont aléatoirement initialisés. L'apprentissage dans cette architecture, le Perceptron, se fait au niveau de ses synaptiques en les modifiant. Dit autrement, si la valeur calculer de Y est différente de celle de la théorie, cas de classification supervisée, alors une mise à jour des poids synaptiques est faite en fonction de l'écart produit entre la sortie calculée et la valeur théorique de Y. Cet apprentissage est connu sous le nom : la loi de HEBB. Ce processus est répété jusqu'à ce qu'on atteint un écart nul ou un nombre maximal d'itérations.

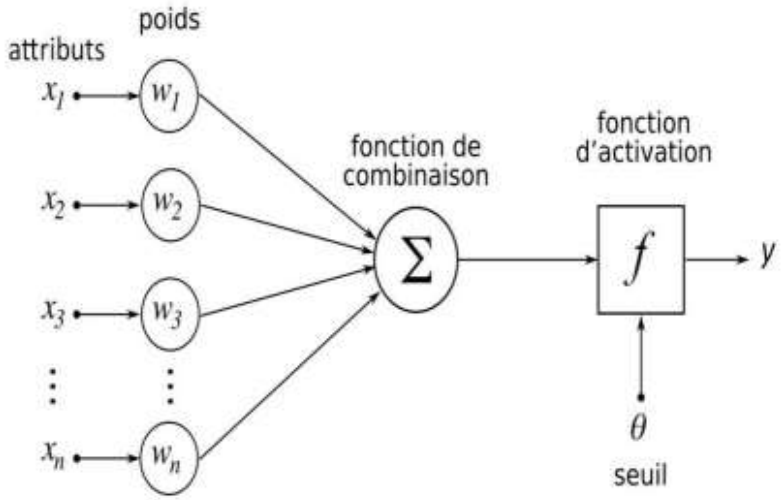


Figure 1-5 un schéma du perceptron

Malheureusement, l'application de ce modèle est strictement limitée à des problèmes simples, notamment ceux de type linéairement séparable, et ne peut être utilisé sur des problèmes complexes réels ou de type non linéairement séparable. Le perceptron Multicouche est une version amélioré du perceptron original dans lequel il existe plusieurs couches cachées. Cette technique lui permet de résoudre toute sorte de problèmes complexes et même de type non linéairement séparables. Le perceptron Multicouche a fait ces preuves dans plusieurs domaines tels que la classification et la reconnaissance de motifs.

1.6.7. La Régression Logistique

La régression logistique est un algorithme de classification utilisé pour attribuer des observations avec des variables réelles à une classe binaire. Elle utilise la fonction sigmoïde logistique pour retourner une valeur de probabilité. L'hypothèse de régression logistique tend à limiter la fonction coût sigmoïde entre 0 et 1.

$$h_0(x) = \frac{1}{1+e^{-(B_0+B_1 X_1+B_2 X_2+\cdots+B_d X_d)}} \quad (1\text{-}6)$$

De plus, La régression logistique modélise le log du rapport des probabilités a posteriori par des modèles linéaires :

$$log\left(\frac{c_1|\, x}{c_2|\, x}\right) = B_0 + B_1 X_1 + B_2 X_2 + \cdots . + B_d X_d \quad (1\text{-}7)$$

1.6.8. Bagging

Les méthodes Bagging (Breiman, 1996), sont généralement conçues pour réduire l'erreur causée par le sur-apprentissage pendant le processus d'apprentissage. Ces méthodes se basent sur la technique d'échantillonnage, (sélection avec remise) dans laquelle on sélectionne de petits sous-ensembles d'observations, échantillons (bootstrap), afin de construire des classificateurs indépendants. Chaque classificateur sera donc entrainé sur un échantillon diffèrent à ceux des autres classificateurs. Les résultats de la classification de différents échantillons sont ensuite combinés et/ou fusionnés pour obtenir le résultat final. La figure ci-dessous explique le concept des méthodes Bagging. Ces derniers, généralement peuvent être utilisées avec n'importe quel type de classificateur et notamment les arbres de décision. L'inconvénient majeur de cette méthode est qu'elle peut parfois conduire à une précision faible à cause des petites tailles d'échantillons d'apprentissage. La méthode Bagging réduit remarquablement le sur-apprentissage mais cela n'est utile que si le classificateur est instable aux petits détails dans la phase d'apprentissage. c.-à-d un petit changement dans les données produit un changement important dans le modèle d'apprentissage généré. Un exemple d'un tel algorithme est l'arbre de décision, qui est très sensible à la façon dont les nœuds supérieurs de l'arbre sont construits dans un espace d'attributs très élevés, (Grandvalet, 2004).

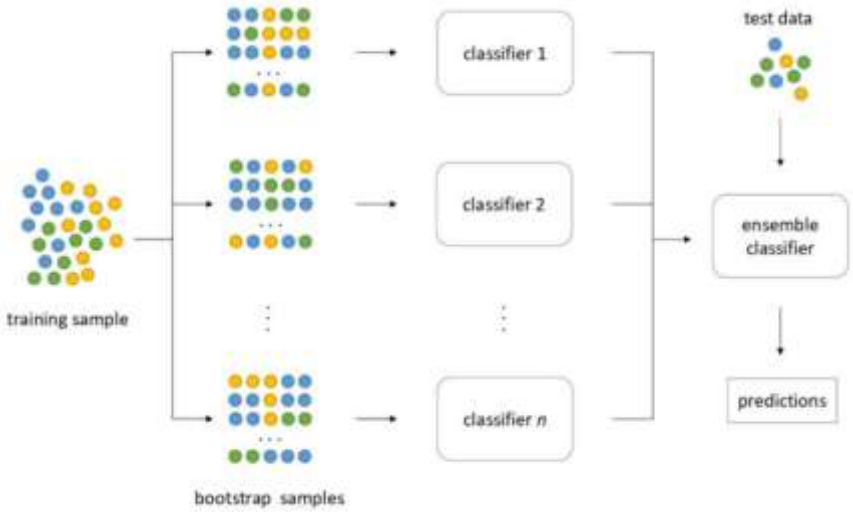


Figure 1-6 : schema general de Bagging

1.6.9. Boosting

Contrairement aux Bagging, les méthodes Boosting se basent sur plusieurs classificateurs faibles mais sur des données construites séquentiellement. Autrement dit, les données d'apprentissage de l'itération i+1 sont générées à partir des données d'apprentissage et les erreurs de classification des i itérations précédentes. De ce fait, les résultats de ces différents classificateurs sont combinés linéairement, où le poids de chaque classificateur dépond de son taux d'erreur de classement. Et à chaque itération on donne plus d'importance aux données mal classées dans les itérations précédentes, cela veut dire qu'on estime dans chaque nouvelle itération de se corriger et les bien classer. AdaBoost est l'algorithme le plus connu utilisant cette technique. La critique majeur de ces méthodes est qu'elles sont très sensibles aux données bruitées, vu qu'ils visent à pondérer les exemples mal classés dans chaque itération, cela peut conduire évidemment à une situation de sur-apprentissage.

Algorithme 1.3 AdaBoost (A=Datat ; c= Algorithme de classification)

d_1 = 1 / *M* % Initialiser le vecteur des poids par 1/M

% M est le nombre d'instances

Pour *t* = 1 à NB. Itération **Faire**

% Trouver un classificateur faible *ht* qui minimise l'erreur selon d_t

h_t= *entrainer_classificateur_faible* (*C, A,* d_t)

e_t = *Erreur*(h_t , *A*) % Taux d'erreur de *ht*

$\alpha_t = \frac{1}{2} ln \left(\frac{1- e_t}{t} \right)$ % Poids du classificateur *ht*

$d_{t+1}(i) = d_t(i) \ exp \left(- \alpha_t \ y_t \ h_t \ (x_t)\right) \ / Z_t$ % Mettre à jour le vecteur d_{t+1}

Fin Pour

Retourner H(x) = Signe ($\sum_{t=1}^{T} \alpha_t h_t \ (x)$)

Dans le pseudo code précèdent. L'algorithme AdaBoost tente dans chaque iteration i d'entrainer un classificateur faible hi. L'erreur de ce dernier fera l'objet d'un coefficient qui lui sera affecté dans la construction du classificateur fort et final (h). Cependant, tous les classificateurs faibles hi sont affectés par des poids d durant leur entrainement. Les instances avec un poids significatif seront prises plus en considération que les autres dans l'étape d'entrainement. La mise à jour des poids est faite itérativement et elle est calculée de sorte que les instances mal classées auront un poids plus important pour que le classificateur se penche sur leurs instances. En conséquent, le prochain classificateur va probablement corriger les instances mal classées. Le Boosting a été appliqué avec pas mal de classificateurs comme, les arbres de décisions (Q. Li, Wu, Wen, & He, 2019), (Q. Li, Wen, & He, 2019). Le classificateur Naïve Bayes (Kim, Hahn, & Zhang, 2000).

1.7. Les collections de textes

L'évaluation de performances des algorithmes de classification textuelle est habituellement effectuée à l'aide des jeux de données, corpus. Connues aussi par Benchmarks, DataSet ou collection. Dans ce chapitre nous allons présenter les trois collections les plus connues et utilisées dans la TC, à savoir Reuters 21-578, 20-New Groups et Ohsumed. Dans ce qui suit on va présenter les définitions de ces collections et la distribution des documents par classe. Toutes ces collections ont été structurées en utilisant le modèle du sac de mots (Salton & Buckley, 1988) dans lequel les documents sont représentés dans un espace vectoriel.

1.7.1. Reuters 21-578

Le corpus Reuters-21578 (D. D. Lewis, 2004), se compose de 21-578 reportages publiés sur le fil de presse Reuters en 1987. Cependant, les documents attribués manuellement aux

catégories ne sont que 12 902. Ces documents sont classés dans 135 catégories. Le découpage ModApte subdivise l'ensemble de données en ensemble de formation et de test de 9603 et 3299 documents, respectivement. Une fois que toutes les catégories des documents dans l'ensemble de test sont cachées, le schéma de classification restant est composé de 90 catégories (R90) et l'ensemble d'entrainement restant se compose de 9 598 documents. Parmi les 90 catégories de R90, nous considérons le sous-ensemble standard composé des 10 plus fréquentes (R10). Il s'agit de la version la plus utilisée, comme confirmé dans (Sebastiani, 2002) page 38. Le tableau 1 montre les détails du contenu de ces dix catégories.

Tableau 1-1 nombre de documents d'entraînements et de testes par catégorie de la base Reuters 21578 ModApte Split.

n°	Class	#train	#test	Total
1	Acq	1650	719	2369
2	Corn	181	56	237
3	Crude	389	189	578
4	Earn	2877	1087	3964
5	Grain	433	149	582
6	Interest	347	131	478
7	Money-fx	538	179	717
8	Trade	369	117	486
9	Ship	197	89	286
10	Wheat	212	71	283
	Total	7193	2787	9980

1.7.2. 20 new groups

La collection de données 20 Newsgroups (20NG), (Lang, 1995b). Cet ensemble de données est une collection d'environ 20.000 documents de groupe de discussion, partitionnés presque uniformément à travers 20 groupes de discussion différents, chacun correspond à un sujet différent. Dans le tableau 1-2, nous présentons un sous-ensemble de la collection 20NG qui comporte 10 groupes de discussion. La répartition des documents est détaillée par groupe (classe) de cette collection sélectionnée.

Tableau 1-2 nombre de documents d'entraînements et de tests par catégorie de la base 20-Newgroups.

n°	Class	# train	# test	Total
1	alt.atheism	480	319	799
2	comp.graphics	584	389	973
3	comp.os.ms-windows.misc	572	394	966
4	comp.sys.ibm.pc.hardware	590	392	982
5	comp.sys.mac.hardware	578	385	963
6	comp.windows.x	593	392	985
7	misc.forsale	585	390	975
8	rec.autos	594	395	989
9	rec.motorcycles	598	398	996
10	rec.sport.baseball	597	397	994
	Total	5771	3851	9622

1.7.3. Ohsumed

La collection de tests OHSUMED est un ensemble de 348 566 références de MEDLINE, la base de données d'informations médicales en ligne, des résumés de 270 revues publiées entre 1987 et 1991 (Hersh, Buckley, Leone, & Hickam, 1994). Un document contient six champs: titre (.T), abstrait (.W), concepts indexés MeSH (.M), auteur (.A), source (.S) et publication (.P). Nous illustrons dans le tableau 1-3. Un échantillon de top 10 catégories de la collection Ohsumed avec distribution des documents par catégorie.

Tableau 1-3 distribution des documents par catégorie dans la collection Ohusmed.

N°	Catégories	# Docs
01	Bacterial Infections and Mycoses	2540
02	Virus Diseases	1171
03	Parasitic Diseases	427
04	Neoplasms	6327
05	Musculoskeletal Diseases	1678
06	Digestive System Diseases	2989
07	Stomatognathic Diseases	526
08	Respiratory Tract Diseases	2589
09	Otorhinolaryngologic Diseases	715
10	Nervous System Diseases	3851
	Total	22813

1.8. Conclusion

La classification automatique des documents textuels est devenue au cours des dernières années un domaine important de recherche. Nous avons tenté dans ce chapitre d'introduire la classification ainsi que ces fondements. Les concepts des algorithmes de classification sont aussi présentés. Ensuite, nous avons décrit les DataSet standards les plus utilisées dans la classification. La performance des algorithmes de classification dépend beaucoup de la qualité de prétraitement des données. Dans cette optique, la phase d'indexation automatique des documents qui est aussi décisive dans le processus de classification sera élaborée dans le chapitre suivant.

Chapitre 2

Indexation des documents textuels

2.1 Introduction

Durant ces dernières années, on est exposé à une quantité massive de documents électroniques qui ne cessent pas de s'accroitre. Malheureusement, la plupart d'entre eux sont non structurés. Par conséquent, l'accès à l'information est très difficile. En outre, l'exploration de ces documents à partir du web est devenue une tâche, de plus en plus lourde d'un côté et coûteuse en termes de temps d'un autre côté. De ce fait, l'exploitation de ces bases documentaires de manière automatique est devenue un besoin primordial. Dans Cette optique, une structuration de ces documents est indispensable afin qu'on puisse appliquer les techniques informatiques, notamment la catégorisation automatique. Habituellement, la structuration de ces documents est une préparation de l'informatisation de ces derniers. On peut distinguer deux phases principales dans leurs structurations, ou indexation, qui sont : le nettoyage et la représentation. Par ailleurs, il existe plusieurs approches de représentation ainsi que différentes techniques de nettoyage dans lesquelles on distingue plusieurs étapes. De plus, chaque type de donnée, à savoir : la vidéo, l'image, la parole et/ou notamment le texte, a son propre processus de prétraitement et ses méthodes de représentation adéquates.

Dans ce chapitre on présente un état de l'art des différentes approches de représentation de textes, les méthodes de pondération des termes et le codage. Ensuite, on introduit les étapes de nettoyage et également un aperçu sur les approches et les méthodes de réduction de la dimensionnalité.

2.2 Prétraitement

Dans tout processus de catégorisation automatique, notamment la catégorisation textuelle, la phase de prétraitement joue un rôle décisif où la performance du classificateur est influencée de façon directe et considérable. C'est-à-dire, un bon nettoyage améliore le résultat de la classification peu-importe la capacité du classificateur appliqué et l'inverse est vrai. Le nettoyage permet éventuellement de dégager tous ce qui est bruit, mot inutile, ainsi que tout mot contribuant de façon négative dans un contexte de catégorisation malgré son importance et son apport dans le texte. Dans cette phase, on devrait habituellement, supprimer les mots inutiles et redondants. En plus, les mots non pertinents à une classe sont aussi éliminés. Cette opération est généralement réalisée à l'aide de mesures statistiques, plus précisément, selon un critère reliant un terme et une classe donnée, à savoir l'information mutuelle, le coefficient de corrélation ou l'entropie entre autres (Karabulut, 2013). En fonction d'un seuil, habituellement défini par l'utilisateur, ces termes sont jugés soit pertinents, soit non porteurs

d'information vis-à-vis d'une catégorie. Par ailleurs, les mots très rares et qui n'apparaissent que dans un taux très faible sur l'ensemble de corpus sont considérés comme des mots inutiles. Évidement on ne peut pas fonder une décision sur un sous ensemble de termes très restreint et par conséquent ils sont dégagés dans le reste du processus d'indexation. Du point de vue informatif, ces mots-là sont classés comme bruit envers l'ensemble des documents malgré leurs importances pour certains documents. A titre d'exemple, la mesure IDF (Inverse Document Frequency) est l'une des mesures appliquées dans ce contexte, elle sert à ressortir, dans un éventuel prétraitement textuel d'un langage naturel, un sous ensemble de termes qui ont une fréquence inférieure à un seuil donné. Les termes très fréquents dans un corpus sont également vus comme des mots non pertinents. Du fait qu'ils figurent dans la plupart des documents, ces termes sont donc considérés non discriminants et leur contribution dans la phase de catégorisation sera purement négative. D'après la loi de ZipF (Zhu, Zhang, Wang, Li, & Cai, 2018) les termes importants sont les termes qui ne sont ni trop fréquents et ni trop rares. En plus des termes fréquents, on en trouve un autre type très particulier qui est les mots vides, ou les mots outils. Ce sont donc, et sans aucun doute, utiles pour toute sorte de langue mais malheureusement sans intérêt dans un contexte de catégorisation. Généralement, dans la phase de prétraitement, une liste préétablie pour chaque langue est utilisée à la suppression de ces mots. À titre d'exemple, une liste de mots vides en français contient les termes suivants : '' le, la, les, ce, ceux, donc, mais, ou ….etc.'. Ces mots sont donc jugés bruit, non porteurs d'information, peu importe leurs qualités statistiques ou sémantiques dans les textes. Il est à noter que ce genre de mots subit un traitement spécifique appelé, l'élimination des mots vides.

2.2.1 Suppression des caractères inutiles

La segmentation (Tokenization) est une étape qui permet de représenter un document par un ensemble de mots, termes (Token) à partir de l'ensemble d'entités linguistiques qui sont apparus dans ce document. Pour ce faire, il faut d'abord savoir déterminer les délimiteurs des mots, autrement dit, les débuts et les fins des mots. En général, comme en Anglais ou en Français, on utilise les délimiteurs suivants : espace, tabulation, virgules…etc. D'un côté, les mots sont ressortis à travers les délimiteurs précédents et d'un autre côté, les délimiteurs seront supprimés. Généralement une liste contenant les signes de ponctuation ainsi que les nombres est utilisée dans cette étape. Par exemple, tous les caractères figurant dans la liste suivante : {.,;:'''()?!&-#0123456789+/<>$^%[]⁄='} seront supprimés, entre temps, cette liste aide à ressortir et de représenter les mots (Token).

"\n\nCOMPUTER TERMINAL SYSTEMS <CPML> COMPLETES SALE\n\n COMMACK, N.Y., Feb 26 - Computer Terminal Systems Inc said\nit has completed the sale of 200,000 shares of its common\nstock, and warrants to acquire an additional one mln shares, to\n<Sedio N.V.> of Lugano, Switzerland for 50,000 dlrs.\n The company said the warrants are exercisable for five\nyears at a purchase price of .125 dlrs per share.\n Computer Terminal said Sedio also has the right to buy\n	"\n\nCOMPUTER TERMINAL SYSTEMS CPML COMPLETES SALE\n\n COMMACK NY Feb 26 Computer Terminal Systems Inc said\nit has completed the sale of 200000 shares of its common\nstock and warrants to acquire an additional one mln shares to\nSedio NV of Lugano Switzerland for 50000 dlrs\n The company said the warrants are exercisable for five\nyears at a purchase price of 125 dlrs per share\n Computer Terminal said Sedio also has the right to buy\

Figure 2-1 Exemple de suppression des caractères inutiles

2.2.2 Élimination des mots vides (stop words)

Les mots vides ou les mots outilles sont éliminés dans cette étape. Dans un corpus, on trouve les mots vides, les mots qui sont les plus fréquents dans ce dernier. D'habitude, on dégage tous ce qui est article, déterminant, adjectif, préposition, conjonction et pronom parmi d'autres. Ces mots-là sont donc non discriminants vu leurs fréquences dans la base textuelle, présents dans tous les documents du corpus. En revanche, leur élimination va assurer un avantage en termes de taille du vocabulaire, réduction, ainsi qu'une amélioration dans les performances des classificateurs. A titre d'exemple, on peut citer quelques mots outilles pour l'Anglais :'the', 'for', 'and', 'a' et 'it'. L'élimination de ces mots n'affecte pas les performances de la catégorisation de textes. Une liste disponible sur le web contenant 571 mots vides est accessible via le lient suivant : (ftp://ftp.cs.cornell.edu/pub/smart/english.stop).

"\n\ncomputer terminal systems cpml completes sale\n\n commack ny feb computer terminal systems inc said\nit has completed the sale of shares of its common\nstock and warrants to acquire an additional one mln shares to\nsedio nv of lugano switzerland for dlrs\n the company said the warrants are exercisable for five\nyears at a purchase price of dlrs per share\n computer terminal said sedio also has the right to buy\n	"\n\ncomputer terminal systems cpml completes sale\n\n commack ny feb computer terminal systems inc said\n completed sale shares common\nstock warrants acquire additional one mln shares \nsedio nv lugano switzerland dlrs\n company said warrants exercisable five\nyears purchase price dlrs per share\n computer terminal said sedio also right buy\n

Figure 2-2 L'élimination des mots vides

"\n\nCOMPUTER TERMINAL SYSTEMS CPML COMPLETES SALE\n\n COMMACK NY Feb 26 Computer Terminal Systems Inc said\nit has completed the sale of 200000 shares of its common\nstock and warrants to acquire an additional one mln shares to\nSedio NV of Lugano Switzerland for 50000 dlrs\n The company said the warrants are exercisable for five\nyears at a purchase price of 125 dlrs per share\n Computer Terminal said Sedio also has the right to buy\	"\n\nCOMPUTER TERMINAL SYSTEMS CPML COMPLETES SALE\n\n COMMACK NY Feb Computer Terminal Systems Inc said\nit has completed the sale of shares of its common\nstock and warrants to acquire an additional one mln shares to\nSedio NV of Lugano Switzerland for dlrs\n The company said the warrants are exercisable for five\nyears at a purchase price of dlrs per share\n Computer Terminal said Sedio also has the right to buy\n

Figure 2-3 La suppression des numéros

2.2.3 Traitement des lettres majuscules

Les lettres majuscules sont très utilisées dans les paragraphes. On les trouve aux débuts des phrases, dans les mots propres et même dans les abréviations. La façon la plus évidente de les traiter est de, tout simplement, les convertir en minuscules. Cependant, cette technique possède des inconvénients et notamment le risque de perte de sens ou d'ambigüité.

A titre d'exemple, le mot 'COMPLETES' devient 'completes'. Par contre, la conversion en minuscule du mot ' IFORMATION TECHNOLOGY' qui s'abrège en anglais à 'IT', produira un résultat conflictuel avec 'it' qui veut-dire en français 'il' ou 'ce', et qui n'a aucun rapport avec le mot d'origine, l'abréviation 'IT'. Pour pallier à ce genre de situation, un traitement supplémentaire et exceptionnel basé sur des dictionnaires particuliers doit être élaboré en parallèle dans le traitement des lettres majuscules.

"\n\nCOMPUTER TERMINAL SYSTEMS CPML COMPLETES SALE\n\n COMMACK NY Feb Computer Terminal Systems Inc said\nit has completed the sale of shares of its common\nstock and warrants to acquire an additional one mln shares to\nSedio NV of Lugano Switzerland for dlrs\n The company said the warrants are exercisable for five\nyears at a purchase price of dlrs per share\n Computer Terminal said Sedio also has the right to buy\n	"\n\ncomputer terminal systems cpml completes sale\n\n commack ny feb computer terminal systems inc said\nit has completed the sale of shares of its common\nstock and warrants to acquire an additional one mln shares to\nsedio nv of lugano switzerland for dlrs\n the company said the warrants are exercisable for five\nyears at a purchase price of dlrs per share\n computer terminal said sedio also has the right to buy\n

Figure 2-4 Le traitement des majuscules

2.2.4 La désuffixation (stemming)

La désuffixation ou le (steming en anglais) est un traitement lexical appliqué sur les mots selon leurs variations morphologiques, à savoir les flexions, dérivations et compositions. Donc elle remplace, ou, regroupe un ensemble de mots de différentes morphologies qui expriment le même sens, par un seul terme qui est leur racine (stem). Le principe de cette opération s'articule sur des règles de troncatures qui servent à éliminer les suffixes des mots, à savoir : 'ED', 'ING, 'ION 'IONS'. A ce titre, en langue anglaise, les termes : ''consult '', ''consultant '', ''consulting '', ''consultantative '' et''consultants '' seront regroupés par le stem ''consult''. Par ailleurs, et pour d'autres termes, l'application de la déssufixation

nécessite un traitement exceptionnel. A titre d'exemple, les mots 'business' et 'busy' se réduisent, en supprimant leurs suffixes, au même mot 'busi' tandis qu'ils ont des sens différents. En plus, un autre inconvénient de cette technique est qu'elle peut fournir des stems qui ne sont pas des morphèmes, c'est-à-dire n'existent pas dans le dictionnaire, comme c'est le cas du mot 'computer' qui devient 'comput'. Par ailleurs, chaque langue a ses propres algorithmes de désuffixation, c'est-à-dire que les règles appliquées en Français ne sont pas les même pour une autre langue. En revanche, les avantages de cette phase sont comme suit : elle ne nécessite pas trop de connaissances linguistiques comparée à la lemmatisation. Aussi, elle ne demande pas un dictionnaire. Les algorithmes de désuffixation sont basés sur des règles de troncature de mots afin de produire des racines, et ils sont, en conséquence, trop rapides. L'algorithme de PORTER (Porter, 1980) est l'un des algorithmes de racinisation les plus reconnus. Il est conçu pour l'anglais et il contient plus de cinquante règles.

"\n\ncomputer terminal systems cpml completes sale\n\n commack ny feb computer terminal systems inc said\n completed sale shares common\nstock warrants acquire additional one mln shares \nsedio nv lugano switzerland dlrs\n company said warrants exercisable five\nyears purchase price dlrs per share\n co-mputer terminal said sedio also right buy\n	"comput termin system cpml complet sale commack ny feb comput termin system inc said complet sale share common stock warrant acquir addit one mln share sedio nv lugano switzerland dlrs compani said warrant exercis five year purchas price dlrs per share comput termin said sedio also right buy

Figure 2-5 Exemple d'application de la racinisation

2.2.5 La lemmatisation

La lemmatisation consiste à remplacer les verbes par leurs formes infinitives, et les noms par leurs formes singulières. Par exemple l'ensemble : { Traveled ; Reflecting ; Moved; workers, Neighborhoods; Plants} est transformé en { Travel ; Reflect ; Move; worker, Neighborhood ; Plant}. En plus, c'est une tâche plus compliquée que la désuffixation dans laquelle on utilise des dictionnaires ainsi que les règles de dérivation afin d'extraire des lemmes ou des racines. TreeTagger (Schmid, 1994) est un algorithme développé dans ce contexte pour les langues anglaise, française, allemande et italienne. A noter que la tâche de

lemmatisation est plus coûteuse en termes de temps et ressources par rapport à la désuffixation mais très compétitive en termes de performances de classification.

2.3 La Représentation de textes

L'application des algorithmes d'apprentissage sur des données de type images, vidéos et notamment sur des données textuelles exige que ces données brutes soient transformées en une forme bien particulière : une forme numérique. Autrement dit, on associe pour chaque donnée en entrée, i.e., document, une donnée numérique à partir de laquelle les techniques d'apprentissages seront applicables. De ce fait, la représentation de textes est une phase essentielle dans un processus de classification de textes où les documents sont représentés par un vecteur de termes. En particulier, dans le modèle de sac de mots les termes sont généralement des mots du corpus. Parmi les différentes méthodes de représentation, on peut citer la représentation binaire où les termes sont représentés avec une valeur binaire qui vaut soit 1 qui indique la présence du terme dans le document, soit 0 qui indique son absence. Une autre alternative de représentation est d'utiliser une valeur entière pour exprimer le nombre d'apparition du terme (la fréquence du terme) pour un document accordé. Par ailleurs, on peut représenter une collection de textes par une matrice d'incidence (Terms Documents Matrix) où les documents représentent les lignes et les colonnes représentent les termes. Plusieurs codages peuvent être utilisés pour la matrice d'incidence, notamment, la fréquence du terme TF, qui exprime le nombre d'occurrence du terme dans le document. Le codage binaire, dans lequel on met 1 pour un document qui contient le terme T et 0, sinon. Le codage TF*IDF est l'un des codages les plus appliqués dans ce contexte. Une simple définition de ce dernier pourrait être : une combinaison des pondérations TF (Term Frequency), et, IDF (Inverse Document Frequency) à la fois. On en trouve plusieurs formules adaptées à ce codage dans la littérature (voir e.g. Sebastiani, 2002). De plus, ce codage tient compte de deux paramètres en même temps : l'un qui désigne l'importance locale (TF) du terme, c-à-d le poids du terme pour un document, et l'autre qui se concentre sur l'importance globale (IDF) du mot, c-à-d le poids du mot vis-à-vis de la collection de textes, ce qui fait de lui un bon indicateur de discrimination.

2.3.1 Représentation des textes par sac de mots

La représentation par sac de mots (bag of words) est l'une des méthodes de représentation les plus connues pour la catégorisation de textes. L'idée clé est de représenter

chaque document par un vecteur de mots, ou termes. Les mots, de leur part sont codés par le nombre d'occurrences dans le document (Salton & McGill, 1986). Dans cette représentation l'ordre des mots est ignoré ainsi que toute analyse grammaticale. En outre, les termes ont un sens explicite. Comme résultat, on a un modèle très simple. En plus, et malgré la détérioration de la structure des textes (perte de l'ordre des mots), cette représentation a montré ses avantages dans plusieurs applications et notamment la catégorisation. Cependant, on trouve une difficulté dans la détermination du terme de sorte qu'il soit traité d'une manière automatique. Dans (Gilli & Conde, 1988) , les auteurs considèrent le terme comme étant une suite de caractères appartenant à un dictionnaire, ou bien, de façon formelle, comme étant une suite de caractères séparés par des espaces ou des caractères de ponctuations. A ce titre, il faut prendre en compte la gestion des mots composés comme court-circuit et un laissez-passer ainsi que les sigles, tel que ERP et CRMI. En plus, il faut étudier le cas des différentes formes morphologiques d'un mot et les grouper en un seul terme. Différentes formes qui comportent le même sens, à savoir : travail, travailleur, travailleurs et travaillons ont des morphologies différentes mais d'un autre côté tous tournent autour d'un sens unique lié au travail. Les regrouper en un seul mot produit non seulement une amplification de la fréquence du terme, ''travail'', mais aussi réduit la taille du vocabulaire. Pour pallier à tous ces problèmes, une phase de prétraitement linguistique est nécessaire.

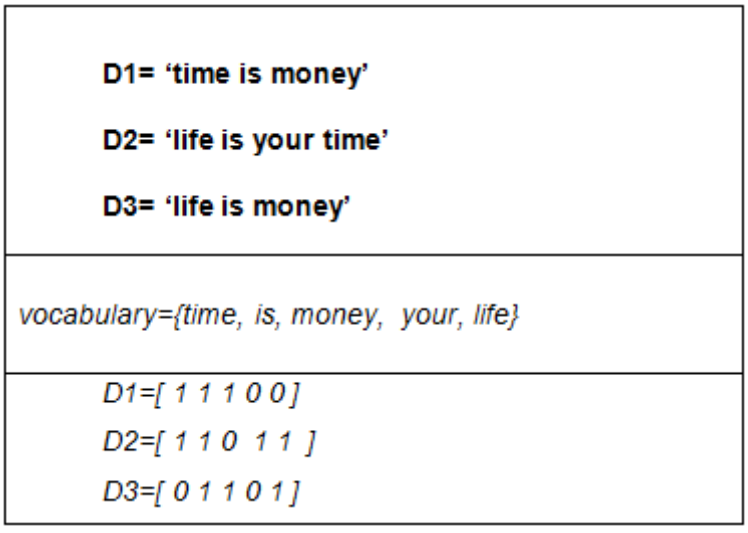


Figure 2-6 La représentation par sac de mots, exemple de trois documents

2.3.2 Représentation en phrases

La représentation par phrases était à l'origine une solution alternative de la représentation de sac de mots afin de conserver le sens des textes. L'idée intuitive c'était qu'une phrase est plus porteuse de sens qu'une représentation à partir des mots distribués dans les textes. C'est-à-dire, avec cette représentation, non seulement on évite la destruction syntaxique produite dans le modèle de sac de mots, mais aussi on peut conserver la sémantique. Logiquement, cette richesse en termes de sémantique devrait conduire à de bons résultats. Néanmoins, les résultats empiriques obtenus en pratique ont montré que les performances des algorithmes de classification étaient nettement faibles en comparaison avec l'approche de sac de mots. Cela est dû au fait que les phrases ont des fréquences très faibles à cause du nombre prohibitif des phrases générées (David Dolan Lewis, 1992). En effet, pour n mots il existe n^k phrases possibles de taille k à générer. Une solution est proposée par (Caropreso, Fernandacnandad, Matwin, & Sebastiani, 2001) pour remédier à cette situation. Les Hauteurs ont utilisé une représentation basée sur des phrases statistiques comme descripteur à la place des phrases grammaticales. Plus précisément, une phrase statistique est une sorte de mots adjacents (pas forcement ordonnés), qui apparaissent ensembles mais qui ne respectent pas forcement les règles grammaticales (Jalam, 2003). Dans (Scott & Matwin, 1999) une étude expérimentale a été effectuée sur une collection de petits textes extraits à partir de Reuters. Cette étude a montré l'avance de l'approche de représentation par phrase (phrase statistique) par rapport à l'approche sac de mots.

2.3.3 Représentation par les N-grammes

La notion de N-gramme a été introduite par (Shannon, 1948). La représentation par les N-grammes consiste simplement à représenter un mot par une suite de caractères qui le composent. Par exemple, si on prenait la phrase 'holly day' alors : hol, oll, lly, ly_, y_d, _da, day seront ses 3-grammes générés. En plus, on peut générer autant de types de n-grammes pour une phrase donnée, à savoir le 2-grammes, le 4-grammes et/ou le 5-grammes. Une définition du n-grammes pourrait être : sur un texte particulier, en faisant déplacer une fenêtre coulissante par un seul caractère, un pas, et à chaque déplacement, le masque capturé par cette fenêtre sera enregistré et considéré comme un n-gramme, i.e., un descripteur. La taille de la fenêtre détermine le n du n-grammes appliqué, c'est-à-dire, si la taille est 2 alors on obtient un 2-grammes…etc. A la fin, l'ensemble de descripteurs enregistrés donne l'ensemble de tous les n-grammes du texte actuel (Miller, Shen, Liu, & Nicholas, 2000).

Les avantages de cette méthode sont multiples. Le grand avantage est qu'elle est indépendante de la langue utilisée contrairement au modèle sac de mots. De plus, cette technique ne nécessite aucun prétraitement ou opérations de nettoyage comme : l'élimination des mots vides, la racinisation, la lemmatisation, etc. vus dans le modèle sac de mots. Même la notion de segmentation a disparu ici. Par conséquent, on n'a plus besoin de déterminer les frontières et/ou les débuts et les fins des mots. Enfin, un autre avantage est la tolérance aux déformations : si on prend un document qui contient des erreurs d'horographes, des mots mal écrits ou scannés avec erreurs, comme c'est le cas des OCR, alors les n-grammes supportent ces erreurs d'horographes et donnent toujours de bonnes performances contrairement à la représentation par sac de mots qui est très sensible aux déformations.

2.4 La pondération

Pour toute représentation de textes, et notamment le modèle sac de mots, il faut choisir une façon efficace de coder les termes d'un vecteur dans lequel un document est représenté. Évidemment, la formule la plus simple est de pondérer les différents mots apparaissant dans les textes par des valeurs égales à 1 ou 0 qui expriment leur présence et leur absence, respectivement. Une autre façon est de les représenter à l'aide de leurs fréquences désignant les nombres d'occurrences des termes pour un tel document.

En général, dans le modèle de sac de mots un vecteur comporte généralement les poids $w_{i,k}$ de chaque mot i pour le document k. Soit la notation suivante :

- $f_{i,k}$: la fréquence du mot i dans le document k.
- N : le nombre de documents dans la collection.
- n_i : le nombre total de fois que le mot i se produit dans la collection.

Certaines techniques de pondération sont décrites dans les sous-sections suivantes.

2.4.1 Pondération booléenne

Dans la formule (2.1) suivante, le poids est égale à 1 si le mot apparaît dans le document et égale à 0, sinon.

$$w_{i,k} = \begin{cases} 1 \quad si \ : \ f_{i,k} > 0 \\ \\ 0 \qquad sinon \end{cases} \qquad 2.1$$

2.4.2 Pondération par fréquence de mot

La formule (2.2) montre la pondération par fréquence qui est tout simplement le nombre d'occurrences du terme pour le document en cours. Elle est exprimée comme suit :

$$w_{i,k} = f_{i,k} \qquad 2.2$$

2.4.3 Pondération TF-IDF

La pondération TF-IDF acronyme de (Term Frequency Inverse Document Frequency) est la formule la plus répandue dans le codage de textes. Elle est issue du monde de la recherche d'information (Salton & Buckley, 1988). Le codage TF-IDF prend en compte deux critères importants à la fois pour un terme : le premier (TF Term Frequency) exprime l'importance locale tandis que le deuxième (IDF : Inverse Document Frequency) exprime l'importance globale, qui est l'inverse du nombre des documents dans lesquelles le terme est apparu. Autrement dit, la pondération TF-IDF tient en compte aussi bien des termes importants dans les documents par la pondération TF que des termes importants par rapport à la collection par la pondération IDF. La formule (2.3) montre les détails de calcul de la mesure TF-IDF. La formulation la plus simple de cette dernière peut être introduite par le produit des deux termes TF et IDF.

Le poids $\boldsymbol{w_{i,k}}$ d'un terme *i* dans un document *K* est calculé par :

$$\boldsymbol{w_{i,k} = TF * IDF = f_{i,k} * log\ (\frac{N}{n_i})} \qquad \textbf{2.3}$$

2.5 Réduction de la dimensionnalité

L'inconvénient majeur de la représentation par sac de mots se présente dans la longueur de son vecteur de caractéristiques. En plus, la majorité de ses caractéristiques sont redondant ou non porteur de sens, bruits. L'élimination des mots vides, en particulier, amorti légèrement la gravité du problème mais elle reste toujours insuffisante. D'autre part, l'impact de la suppression des mots rares dans le corpus reste également faible. Pour y remédier, une étape supplémentaire de réduction de dimensionnalité est indispensable. Elle sert à réduire la taille du vocabulaire en dehors de la phase de prétraitement, en éliminant certains attributs avant d'appliquer les techniques de classification. Cela permet, à la fois, d'accélérer le temps de calcul de la fonction de classification, réduire l'espace de stockage et éviter les problèmes de sur-apprentissage résultant de la taille du vocabulaire, ce qui améliore les performances de la phase de classification.

On distingue deux grandes familles qui utilisent des techniques issues soit de la théorie de l'information soit de l'algèbre linéaire (Sebastiani, 2002) : i) selon qu'elles agissent localement ou globalement. Pour le cas de réduction locale de dimension, étant donnée une catégorie $\boldsymbol{c}$, on choisit un sous ensemble de termes avec lesquels les documents seront représentés. Cependant, chaque document aura un sous ensemble de termes diffèrent en fonction des catégories. Quant à la réduction globale de dimension, on ne fait ressortir qu'un seul sous ensemble de termes. Ce dernier sera valable et unique pour toutes les catégories, ce qui fait qu'un document est représenté par un seul sous-ensemble d'attributs dans toutes les catégories. ii) Selon la nature des résultats de la sélection (s'agit-il d'une sélection de termes ou d'une extraction de termes ?). Il existe ici deux approches. L'approche de sélection d'attributs (*feature selection*) qui prend les attributs d'origine, garde seulement ceux jugés utiles à la classification selon un critère et rejette les autres attributs. On trouve plusieurs mesures statistiques dans ce contexte, à savoir : la fréquence, le Gain d'information, l'information mutuelle, la mesure chi square X^2 et la force du terme, entre autres.

La deuxième approche qui est l'extraction d'attributs (feature extraction) génère à partir des attributs de départ de nouveaux attributs en faisant soit des regroupements ou des

transformations. L'analyse en composantes principales (ACP) et l'analyse sémantique latente (LSA) sont parmi les méthodes relevant de cette approche.

2.5.1 Extraction de caractéristiques

Pour l'extraction de caractéristiques (Gomez, Boiy, & Moens, 2012), l'espace d'entités d'origine est converti en un nouvel espace plus compact. Toutes les caractéristiques originales sont transformées dans le nouvel espace réduit sans les supprimer mais en les remplaçants par un ensemble représentatif plus petit.

2.5.1.1 Analyse des composants principaux (PCA)

L'analyse en composantes principales (PCA) est une technique bien connue dans le domaine d'analyse descriptive dans les statistiques. L' PCA permet de réduire la dimensionnalité des données en transformant l'espace d'origine des fonctionnalités en un espace plus petit. C'est-à-dire que l'analyse en composantes principales permet d'extraire de nouvelles variables non corrélés entre eux à partir des variables d'origine qui sont corrélés. Cela est obtenu en transformant les variables d'origine $Y = [y_1, y_2, ..., y_p]$ par un nouvel ensemble de variables, $F = [f_1, f_2, ..., f_p]$ qui sont des combinaisons des variables d'origine. Le processus de calcul repose généralement sur trois étapes : en calculant d'abord la moyenne (μ) de l'ensemble de données, après cela, on passe au calcul de la matrice de covariance des attributs d'origine (Verbeek, 2000), et en troisièmes lieu, finalement, on extrait les vecteurs propres à partir de cette matrice de covariances. Les vecteurs propres ou les composantes principales sont instaurés comme étant une transformation linéaire de l'espace d'attributs d'origine à un nouvel espace dans lequel les attributs ne sont pas corrélés. Les vecteurs propres peuvent être triés selon le critère de la quantité de variation des données originales, appelé aussi inertie. Les meilleurs vecteurs propres, ceux avec la plus haute valeur propre, sont sélectionnés comme nouvelles fonctionnalités et les autres sont jetés.

2.5.1.2 Analyse sémantique latente (LSA)

L'analyse sémantique latente (LSA), en Anglais *: Latent semantic analysis*, est une méthode de traitement des langues naturelles. Généralement, LSA analyse les relations entre un terme et les concepts contenus dans une collection non structurée de textes. Cette méthode a la capacité de corréler des termes sémantiquement liés et qui sont cachés dans un texte. Précisément, LSA produit un ensemble de concepts à partir de la matrice d'occurrences.

L'espace de concepts est plus petit que l'ensemble d'origine de termes. Le plus important ici, est que ces concept sont liés à la fois aux documents et aux termes (Xia & Wong, 2006), (L'Huillier, Hevia, Weber, & Ríos, 2010). En outre, LSA utilise une décomposition en valeurs singulières (SVD) pour identifier le motif entre les termes et leurs concepts contenus dans le texte, ce qui permet par la suite d'extraire des relations entre les documents. L'analyse sémantique latente est particulièrement utilisée pour les systèmes de récupération de pages ainsi que dans le regroupement de textes (*text clustering*).

2.5.2 Sélection de caractéristiques

La sélection des caractéristiques, en anglais : *Feature Selection* (FS), contrairement à l'extraction des attributs nous permet de ressortir des sous-ensembles à partir de l'ensemble d'origine. On trouve plusieurs applications de la sélection des caractéristiques dans différents domaines tels que la catégorisation de textes (Lee & Lee, 2006), le regroupement de documents (Kogan, 2003), le traitement d'image, la vision par ordinateur, la bio-informatique et les applications industrielles (Bogunović, 2015). La figure ci-dessous montre le processus général de sélection d'attributs.

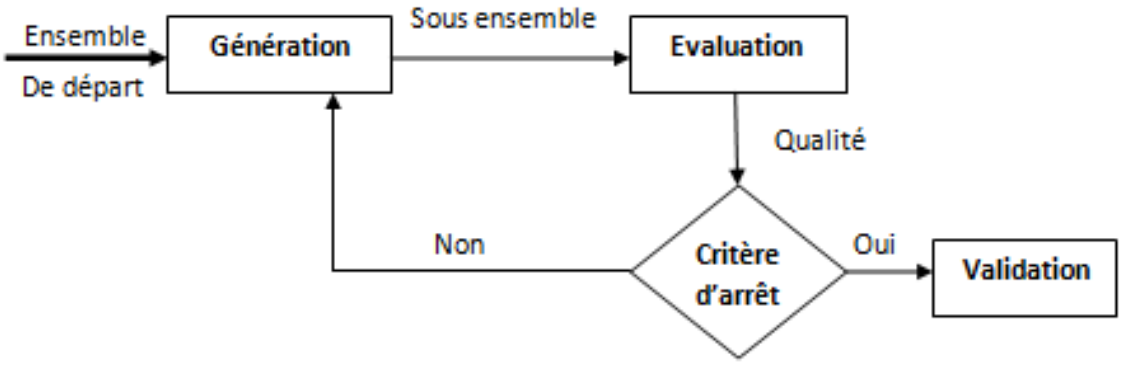


Figure 2-7 processus de sélection d'attributs

2.5.2.1 Le choix d'attributs

La tâche de classification est influencée directement par l'ensemble de termes sélectionnés résultants aussi bien de la phase de prétraitement que de la phase de sélection. L'élimination des caractères intitules et redondants va sûrement améliorer le comportement des classificateurs. Autrement dit, si le terme est pertinent alors on aura une bonne performance, et dans le cas contraire (non pertinence du terme) la performance sera dégradée.

Plusieurs définitions de pertinence existent dans la littérature. Dans (John et al. [1994], John [1997]), les auteurs stipulent qu'il existe trois types de termes: très pertinent, peut pertinent et non pertinent.

- **Très pertinent** : Un terme t_i est dit très pertinent si son absence cause une dégradation significative de la performance du classificateur.
- **Peu pertinente** : Un terme t_i est dit peu pertinent s'il n'est pas "très pertinent" et s'il existe un sous-ensemble *V* tel que la performance de $V \cup \{t_i\}$ soit nettement meilleure que celle de *V*.
- **Non pertinente** : Les attributs qui ne sont ni "peu pertinents" ni "très pertinents" représentent les attributs non pertinents. Ces attributs seront en général supprimés de l'ensemble d'attributs originaux.

Pour l'évaluation des attributs, les procédures utilisées dans les algorithmes de sélection peuvent être classées en trois catégories principales : "Filter", "Wrapper" et "Embedded". Dans la suite, on va introduire chacune de ces catégories en montrant leurs propriétés.

On ce qui concerne le critère d'arrêt dans les méthodes Filter on définit un seuil pour le nombre d'attributs retenus après avoir calculé le score de chacun des attributs. Pour les deux autres méthodes, Wrapper et Embbeded, le processus de recherche est arrêté si on atteint un nombre maximal d'itérations ou si on n'arrive plus à obtenir une amélioration dans les sous-ensembles générés.

2.5.2.2 Filter

Dans la sélection d'attributs, l'approche *Filter* a été la première approche appliquée dont le critère d'évaluation est la quantification de la pertinence d'un attribut, seulement, à partir des propriétés des données d'apprentissage. Cette méthode est considérée, davantage comme une partie réservée au processus de prétraitement. En plus, l'évaluation se fait généralement indépendamment de l'algorithme de classification, (Yiming Yang & Pedersen, 1997).

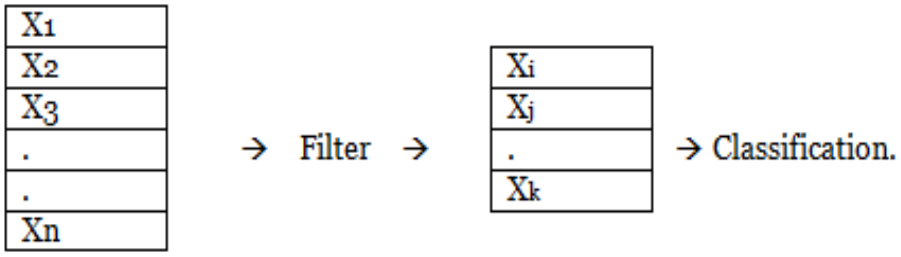


Figure 2-8 schéma de l'approche 'Filter'

Selon la figure 2-8., il est claire que l'objectif de cette approche est de sélectionner un sous ensemble réduit d'attributs qui ont un score supérieur au seuil définit par l'expert. Tous les attributs de ce sous ensemble seront considérés comme pertinents et vont être sollicités en phase d'apprentissage. Par la suite, nous présentons quelques méthodes utilisées dans la littérature comme score ou critère d'évaluation dans cette approche.

2.1.1.1.1 La fréquence

La fréquence, ou, *Document Frequency* (DF) en anglais, est une mesure dans laquelle on supprime ou on élimine tout simplement les mots qui sont apparus dans la plupart des documents de la collection en cours. Généralement, un seuil défini par l'utilisateur servira ultimement à éliminer ceux qui ont un score (fréquence) supérieur à ce dernier. Généralement les termes fréquents dans la collection sont non informatifs. De ce fait, ils sont inutiles pour la catégorisation. D'autre part, il est possible que certains termes soient le résultat d'erreurs, comme un mot mal orthographié. Dans ce cas, leur élimination peut contribuer poussivement dans la classification. Cependant, l'inconvénient majeur de de cette technique est que généralement, elle élimine des termes avec une fréquence faible ou moyenne malgré leurs importances dans les textes.

2.1.1.1.2 La force du terme

La force du terme ou *Term Strength* en anglais, permet d'estimer l'importance d'un terme en fonction de sa propension à apparaître dans des documents semblables. D'abord, on commence à former des paires de documents dont la similarité basée cosinus est supérieure à un certain seuil. Ensuite, la force d'un terme sera calculée à l'aide de la probabilité

conditionnelle qu'il apparaisse dans le deuxième document d'une paire, sachant qu'il apparaît dans le premier.

2.1.1.1.3 Le coefficient de Gini

Le coefficient de Gini ou l'indice de Gini, en anglais Gini-Index est une mesure statistique qui calcule le degré de discrimination d'une caractéristique vis-à-vis d'une catégorie. En outre, l'indice de Gini est très connu comme méthode de sélection de caractéristiques pour la classification automatique de textes. La formule ci-dessous montre les détails du calcul.

$$\boldsymbol{Gini(t, c_i) = \sum_{i=1} p(t|ci)^2 . p(ci|t)^2} \qquad \textbf{2.4}$$

Sachant que $\mathrm{p}(\mathrm{t}|c_i)$ et $\mathrm{p}(c_i|\mathrm{t})$ sont la probabilité de *t* sachant que la catégorie $c_i$est présente, et la probabilité de c_i sachant que le terme *t* est présent, respectivement.

2.1.1.1.4 Le gain d'information (information Gain)

Le gain d'information, en anglais : information gain, mesure le nombre de bits d'information obtenus pour la prévision par catégorie en connaissant la présence ou l'absence d'un terme dans un document. La mesure IG d'un terme *t* et d'une catégorie c_i est donnée par la formule (2.5). Pour simplifier la formulation IG, nous introduisons les quatre tuples de dépendance suivants :

- (t, c_i) : la présence de *t* et l'appartenance à c_i.
- $(t, \bar{c}_i)$: la présence de *t* et non-membre dans c_i.
- $(\bar{t}, c_i)$: l'absence de *t* et l'appartenance à c_i.
- $(\bar{t}, \bar{c}_i)$: l'absence de *t* et non-membre dans c_i.

$$\mathbf{IG(t, c_i) = \sum_{c \in \{c_i|\bar{c}_i\}} \sum_{t' \in \{t_i|\bar{t}_i\}} p(t', c). \log \frac{p(t', c)}{p(t')p(c)}} \qquad \textbf{2.5}$$

Sachant que *p* est la probabilité qu'un terme soit présent ou absent dans une classe donnée. Cette probabilité représente le nombre d'occurrences du terme *t* dans les documents appartenant à la classe c_i.

2.1.1.1.5 L'information mutuelle

L'information mutuelle, en anglais : *mutual information*, permet de mesurer la quantité de dépendance, corrélation, entre deux variables aléatoires, dans notre cas un terme et une catégorie donnée. Elle se base sur la théorie des probabilités et a une relation avec la mesure d'entropie. Typiquement, l'information mutuelle entre un terme *t* et la classe *c*, noté *M(t,c),* est la quantité d'information moyenne obtenue sur *t* en observant *c*. l'inconvénient de cette mesure est qu'elle favorise les termes rares à l'opposé des termes fréquents pour une même probabilité conditionnelle (Réhel, 2005). Cette mesure a été fréquemment utilisée pour la sélection de caractéristiques dans la catégorisation de textes (voir e.g. (David Dolan Lewis, 1992) , (Dumais, Platt, Heckerman, & Sahami, 1998)).

$$\boldsymbol{MI(t,c) = log\ \frac{P(t,c)}{P(t).p(c)}} \qquad \textbf{2.6}$$

On note que p(t|c) est la probabilité de *t* sachant que la catégorie c_i est présente, tandis que P(t) et p(c) représentent les probabilités du terme t et la catégorie c respectivement.

2.1.1.1.6 La statistique du X^2

La mesure statistique X^2, en anglais Chi-square(CHI), est la mesure statistique qui s'ajuste bien à la sélection de caractéristiques en évaluant le manque d'indépendance entre un terme et une catégorie donnée. Elle se base sur les mêmes notions de cooccurrence que l'information mutuelle, nombre d'apparition du mot par catégorie, mais une différence importante est qu'elle est soumise à une normalisation, qui rend plus comparable les termes entre eux. A noter que le problème de termes peu fréquents, (rares) persiste toujours comme dans le cas de l'information mutuelle (Y Yang & Pedersen, 1997).

$$X^2(t, c_i) = \frac{N*(AD-CB)^2}{(A+C)(B+D)(A+B)(C+D)} \qquad 2.7$$

Où les termes de l'équation précédente signifient :

- N : le nombre de documents du corpus.
- A : le nombre d'apparitions de t et c_i en même temps.
- C: le nombre de présences de t et d'absences de c_i.
- B : le nombre d'absences de t avec l'absence de c_i.
- D : le nombre d'absences de chacun de t et c_i en même temps.

2.5.2.2.1 Limite des méthodes 'Filter'

Malgré leur succès, leur efficacité, et même si on les trouve dans plusieurs domaines d'applications, la performance de ces dernières reste toujours très limitée en comparant avec les autres approches : 'wrapper' er 'Embeded'. La cause principale de cette limite est que les méthodes basées 'Filters' ne tiennent pas en compte l'influence des caractéristiques sélectionnées sur la performance du classificateur utilisé par la suite. Ainsi, elle ne prend pas en considération la redondance entre les caractéristiques mais elle se contente d'une évaluation individuelle des attributs avec le vecteur classe. Justement, l'avance des autres approches est due à la prise en compte de ces deux points.

2.5.3 Wrapper

Les méthodes enveloppantes ou "Wrapper", en anglais se basent généralement sur un algorithme de recherche pour bien explorer l'espace de solutions, et un algorithme de classification pour l'évaluation des sous-ensembles sélectionnés. A la fin de la recherche, le meilleur sous-ensemble sera adopté comme une solution finale. Evidemment, on constate que cette approche requiert trop de ressources et est également très coûteuse en termes de temps. Un autre inconvénient est que l'algorithme de classification utilisé dans l'étape de classification finale est très dépendant de celui utilisé dans étape de classification. L'approche Wrapper est illustrée dans la figure en bas :

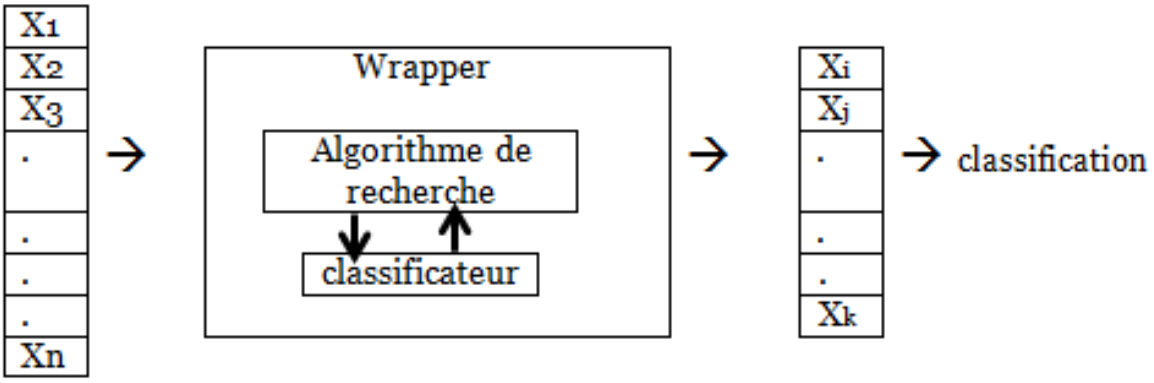


Figure 2-9 : schéma général d'un Wrapper

2.5.4 Embedded

Les méthodes 'Embedded', appelées aussi les méthodes intégrées, incorporent la sélection de variables lors du processus d'apprentissage. Plus particulièrement, dans les méthodes Wrapper on trouve que la base d'apprentissage de départ est divisée en deux parties: une base d'apprentissage et une base de validation pour valider les sous-ensembles des caractéristiques sélectionnés. En revanche, les méthodes 'Embedded' intégrées peuvent se servir de tous les exemples d'apprentissage pour générer le sous ensemble optimal, ce qui peut améliorer les résultats. Un autre avantage de ces méthodes est leur plus grande rapidité par rapport aux approches "Wrapper" parce qu'elles évitent que le classificateur refasse complètement sa tâche pour chaque sous-ensemble de caractéristiques.

2.6 Conclusion

Dans ce chapitre nous nous sommes penchés sur la numérisation des documents textuels. Cette phase est non seulement primordiale mais également décisive pour l'informatisation des données. En premier lieu, on a introduit les étapes de nettoyage. Ensuite, on a présenté un état de l'art des différentes approches de représentation de textes, les méthodes de pondération des termes et également les différents codages de termes. La deuxième partie de ce chapitre a été consacrée au problème de grande dimensionnalité. Dans cette partie nous avons présenté un aperçu sur les approches et les méthodes de réduction de la dimensionnalité. Les méthodes existantes présentent des limitations : au niveau de la complexité pour les méthodes' Warraper' et en ce qui concerne les approches 'Filter', l'inconvénient réside dans la possibilité de sélection d'attributs redondants et inter-corrélés. Maintenant que nous avons ébauché les approches de réduction de la dimensionnalité, nous

poursuivons dans le chapitre suivant avec la présentation des principaux algorithmes de recherche transposés dans ce domaine.

Chapitre 03

Les algorithmes de recherche et sélection d'attributs.

3.1 Introduction

La sélection des attributs joue un rôle primordial dans n'importe quel processus d'apprentissage et notamment la catégorisation textuelle. Elle permet d'extraire un sous-ensemble non seulement optimal mais également pertinent. Cela influence positivement sur la qualité de Précision et du Rappel des algorithmes de classification. Dans le chapitre précèdent on a instauré les concepts ainsi que les différentes approches utilisées pour la sélection d'attributs. En continuant dans cette optique, ce chapitre traite principalement les algorithmes de recherche qui représentent le noyau du processus de FS. Autrement dit, la qualité du sous ensemble sélectionné dépend fortement de la performance de l'algorithme de recherche. Dans les sections suivantes, nous allons introduire les différentes stratégies de recherche. Des algorithmes de sélection des attributs et notamment des algorithmes de recherche stochastiques seront présentés et décrits. En outre, nous allons introduire les concepts, les particularités ainsi que les caractéristiques de chacun des algorithmes présentés.

3.2 Les méthodes de sélection

Les méthodes de sélection comportent en général quatre étapes dont l'objectif est d'extraire le meilleur sous ensemble de caractéristiques. Ces étapes sont exprimées comme suit : Initialisation, procédure de recherche ou de génération de solutions, une méthode d'évaluation et finalement le critère d'arrêt, (Liu & Yu, 2005). Une méthode de sélection démarre tout d'abord par une solution initiale, la procédure de recherche permet donc de générer de nouvelle solutions à partir de celle-ci, où la méthode d'évaluation de son côté servira à l'attribution d'un score à chacune des solutions trouvées. Ces deux étapes sont répétées itérativement jusqu'à la satisfaction du critère d'arrêt. Dans la section suivante on met l'accent sur les procédures de recherche en décrivant leurs propriétés ainsi que leurs catégories de recherche.

3.2.1 Etat de l'art

Dans la littérature, il existe un certain nombre de méthodes de FS de type Filtre basées sur des métriques de classement des attributs. Dans le contexte de la catégorisation du texte, les auteurs (Yang et Pedersen, 1997) ont comparé plusieurs méthodes telles que la fréquence des documents (DF), le gain d'information (IG), l'information mutuelle (MI), les métriques Chi-Square (X^2) et la Force du Terme pour FS. Cette comparaison a montré que IG et X^2 sont les plus efficaces. Par ailleurs, Forman (George Forman, 2003) a présenté une

comparaison empirique étendue de douze méthodes de sélection de caractéristiques. Il a constaté que la métrique de sélection d'attributs appelée « Séparation Bi-Normal » (BSN) dépasse toutes les autres méthodes y compris les mesures IG et $X^{2.}$

Du point de vue complexité, la sélection des attributs est un problème difficile. Pour un espace de recherche donné de taille N, il existe 2^N combinaisons possibles de sous-ensembles d'éléments (Mladeni, 2006). Généralement, les algorithmes FS comprennent des stratégies de recherche heuristiques ou aléatoires afin d'éviter cette complexité prohibitive. Cependant, le degré d'optimalité du sous-ensemble de caractéristiques final est souvent réduit. La sélection des attributs a trouvé des applications dans de nombreux domaines, y compris la catégorisation textuelle (Lee & Lee, 2006), le regroupement de documents (Kogan, 2003), le traitement d'images, la vision par ordinateur, la bio-informatique et aussi les applications industrielles (Bogunovic, 2015), entre autres. On outre, une famille d'algorithmes stochastiques : tels que les GAs, les algorithmes basés sur les essaims de particules, notamment PSO et ACO. Ces méthodes sont très couramment utilisées dans plusieurs domaines d'optimisation et notamment pour la sélection des caractéristiques. Au sujet de la catégorisation de textes, on trouve plusieurs métaheuristiques qui ont été proposées dans le contexte de sélection des attributs, à savoir: les algorithmes génétiques combinés avec des méthodes classiques de FS, tel que le gain d'information (Uğuz, 2011), (Ghareb, Bakar et Hamdan, 2016). Un algorithme d'optimisation amélioré des colonies de Fourmis est proposé pour la sélection des termes dans (Janaki Meena, Chandran, Karthik et Vijay Samuel, 2012). Des algorithmes d'essaim particuliers de type PSO ont été également proposés pour la sélection des termes (Baykan, 2014; Chantar et Corne, 2011), (Karabulut, 2013).

Dans l'approche Envellope (Wrapper), un certain nombre de métaheuristiques, algorithmes de recherche, ont été utilisés pour résoudre le problème de sélection d'attributs pour TC. Les algorithmes génétiques (GA) sont une sorte d'algorithmes évolutifs, basés sur l'inspiration naturelle (Holland & Reitman, 1978). Ils imitent les opérations de mutation, de sélection et de reproduction des gènes dans la nature. Par exemple, les GA ont été utilisés comme un outil pour résoudre le problème de sélection d'attributs dans le contexte de l'exploration de données (Siedlecki et Sklansky, 1989). L'optimisation par colonies de fourmis (Ant Colony Optimization (ACO)) est une approche métaheuristique appartenant à la famille des méthodes d'essaims intelligentes. ACO est une méthode de recherche aléatoire

basée sur le comportement alimentaire des essaims (fourmis) (DORIGO, 1992). Dans (Aghdam, Ghasem-Aghaee et Basiri, 2009), les auteurs ont appliqué une méthode de FS basée sur l'algorithme ACO pour la catégorisation de texte. En outre, (Al-Ani, 2005) propose une méthode de FS avec l'algorithme BCA qui est appliquée au problème de la classification de la parole.

L'algorithme Sinus Cosinus (SCA) a été proposé par Mirjalili (2016). SCA est un algorithme d'optimisation globale qui utilise un modèle mathématique basé sur les fonctions sinus et cosinus pour mettre à jour de manière itérative un ensemble de solutions candidates. L'efficacité de l'algorithme SCA a été prouvée dans plusieurs applications, notamment les fonctions dans l'espace continu, où SCA a été testé sur plusieurs jeux de données. SCA a aussi fait ses preuves dans la résolution de problèmes réels tels que le problème de la conception aérodynamique (Mirjalili, 2016). De plus, (Abd El Aziz, Selim et Xiong, 2017) ont implémenté SCA pour la détection de galaxies à l'aide de la récupération d'images. Récemment, un algorithme amélioré de SCA, a été proposé pour la résolution des problèmes d'optimisation globale (Abd Elaziz, Oliva et Xiong, 2017). Cette variante de l'algorithme SCA considère l'apprentissage basé sur l'opposition (OBL_SCA) en tant que mécanisme pour la stratégie de diversification afin de mieux explorer l'espace de recherche. Les auteurs ont validé leur proposition en faisant différents tests sur plusieurs bancs d'essai contenant des fonctions continues, ainsi que, sur certains problèmes d'ingénierie afin de mieux évaluer sa capacité de résolution des problèmes réels. Dans (Li, Li et Deng, 2017), les auteurs proposent aussi un algorithme amélioré de sinus cosinus basé sur le prélèvement. L'idée principale, est la suivante : les individus qui n'ont pas réussi à marquer de progrès vont mettre à jour leur position en utilisant les paramètres des variable Levy-Fly dans le but d'améliorer la capacité d'exploration. Cet algorithme a montré de bonnes performances dans le problème complexe d'optimisation non linéaire. Un nouveau mécanisme pondéré de position de mise à jour visant à améliorer les performances de l'algorithme sinus cosinus (Weighted_SCA) a été proposé dans (Meshkat & Parhizgar, 2017) . Il a été démontré que l'algorithme Weighted_SCA est meilleur que l'algorithme SCA, et en mesure de converger plus rapidement que SCA. Récemment, et dans l'application d'optimisation globale, plusieurs algorithmes ont été proposés, notamment : Une synergie de l'algorithme sinus-cosinus et de l'optimisation par essaims de particules (Nenavath, Ravi Kumar Jatoth et Das, 2018), ainsi qu'un algorithme amélioré sinus-cosinus basé sur des informations

parallèles orthogonales (M. Rizk-Allah, 2018). En outre, les auteurs dans (Zhang, Zhou et Luo, 2018) proposent un algorithme amélioré pour l'optimisation basé sur la vague d'eau et l'algorithme SCA : Water Wave Sine Cosine algorithm .

Concernant l'algorithme SCA et son application pour les méthodes FS, SCA a été utilisé comme un système pour la sélection des attributs dans (Hafez, Zawbaa, Emary, et Hassanien, 2016). Dans ce travail, l'algorithme SCA a été implémenté et testé sur 18 jeux de données. Les expériences montrent une avance de SCA par rapport aux autres méthodes de recherche qui utilisent les algorithmes PSO et GA. Finalement, une autre version améliorée de l'algorithme SCA couplée avec une stratégie élitisme a été appliquée pour FS dans l'apprentissage automatique (Sindhu, Ngadiran, Yacob, Zahri, & Hariharan, 2017). Dans cette variante, une mise à jour pour modifier la position des solutions a été proposée. cependant cette solution requiert trop de paramètres supplémentaires qui devraient être ajustés.

3.3 Les algorithmes de recherche

Les méthodes de sélection, peuvent être réparties en trois approches principales basées sur la procédure d'évaluation : Filter, Wrapper et approche Embedded (Gheyas & Smith, 2010), sachant que dans l'approche Filtre, le critère d'évaluation est indépendant de l'algorithme d'apprentissage. Dans l'approche Enveloppe (W*rraper*), les méthodes de FS utilisent un algorithme d'apprentissage pour évaluer un sous ensemble de variables. Cependant, La troisième approche 'Embedded', les méthodes 'Filters' sont imbriqués dans la méthode Wrapper.

Le choix d'algorithme de recherche est décisif dans la méthode de sélection et sa performance influe sur la qualité du sous-ensemble sélectionné. Nous distinguons trois types d'algorithmes de recherche : les algorithmes exponentiels, les algorithmes séquentiels et les algorithmes aléatoires. Nous illustrons en bas une classification de ces derniers. Les algorithmes exponentiels, déterministes, évaluent un certain nombre de sous-ensembles qui augmentent de façon exponentielle par rapport à la taille de l'espace de recherche. En revanche, on obtient un sous-ensemble optimal (idéal). Par ailleurs, les algorithmes séquentiels sont des heuristiques qui ajoutent ou suppriment des variables de manière séquentielle, ce qui les rend confrontés aux problèmes de la convergence prématurée vers des optimums locaux. Enfin, les algorithmes aléatoires considèrent les problèmes de FS comme des boîtes noires. Ces derniers algorithmes atteignent la valeur de la variable de sortie de manière itérative en modifiant les valeurs des variables d'entrées dans leur procédure de

recherche. Ce qui leur permettra d'éviter les optimums locaux et les rend également bien adaptés à tout type de problème d'optimisation.

3.3.1 Les stratégies de recherche

La procédure de recherche permet la mise à jour des solutions trouvées afin d'améliorer le résultat de la fonction objectif. En fait, la qualité de la procédure joue un rôle très important dans la recherche d'un sous-ensemble optimal. Cette dernière devrait être prédéfinie avant le démarrage du processus de sélection des attributs. En général, les stratégies de recherche peuvent être classées en trois catégories : exhaustive, heuristique et stochastique.

- ***Exhaustive***

  Une recherche est dite exhaustive si elle fait le balayage de tout l'espace de recherche, en conséquent le meilleur sous-ensemble est retourné. L'inconvénient de cette approche est que le nombre de combinaisons croit exponentiellement en fonction du nombre de caractéristiques, (Mladeni, 2006) . Pour N attributs, 2^N combinaisons possibles de sous-ensembles. Pour le cas de classification des documents où le nombre de termes est communiqué en centaines de milliers, l'application de cette approche devient impossible.

- ***Heuristique***

  Une stratégie alternative à la précédente est la stratégie de recherche heuristique. Cependant, le degré d'optimalité du sous-ensemble de caractéristiques final est souvent réduit. L'algorithme de recherche ici est itératif. Dans chaque itération on fait ajouter ou rejeter un/des caractéristiques à l'ensemble de caractéristiques en cours. Cette approche se caractérise par sa simplicité et rapidité mais elle tombe souvent dans l'optimum local. Dans la littérature, les trois sous-catégories les plus connues de cette approche sont : Forward, BackWard et StepWise.

- ***Stochastique***

  Pour remédier au problème d'optimum local des méthodes heuristiques, une solution alternative est d'utiliser les algorithmes stochastiques. Ce dernier type se base sur l'aléatoire, que ce soit, dans la génération de la population initiale et/ou également dans la modification des solutions actuelles. Ce comportement aléatoire le rend de

plus en plus non-déterminé. Ce qui lui donne une capacité de recherche exceptionnelle et impressionnante vis-à-vis des deux autres stratégies vues précédemment. La figure 3.1 montre une liste d'algorithmes à titre d'exemple uniquement.

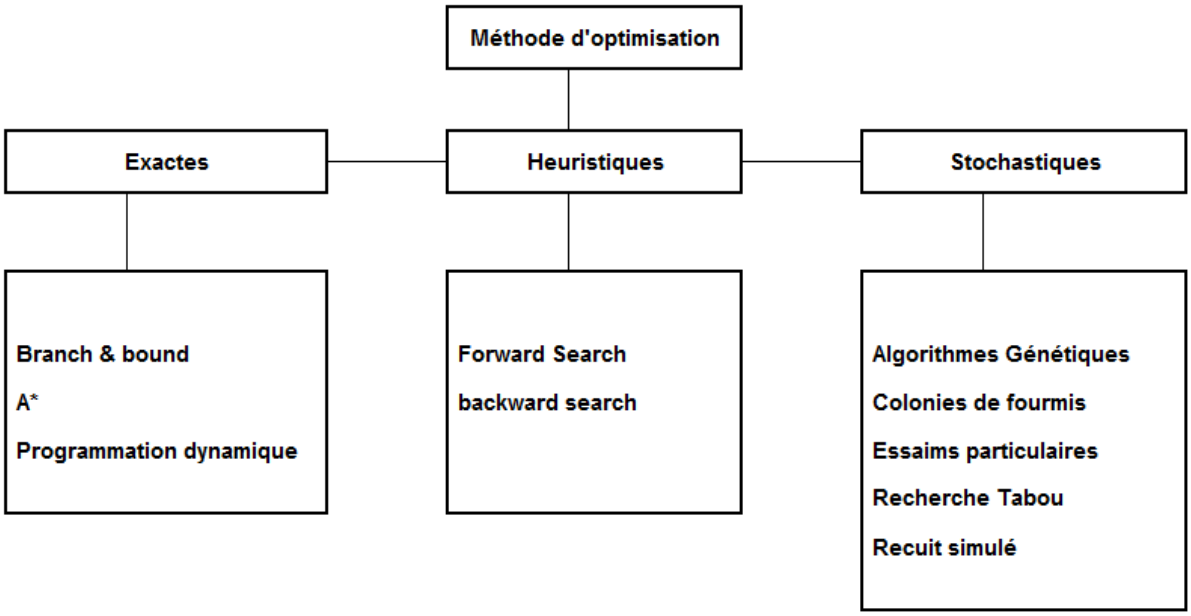


Figure 3-1 classification des algorithmes de recherche

Dans la suite, nous détaillerons certaines méthodes, dans le but d'avoir une vue globale des algorithmes de recherche existants. Parmi eux on trouve des stratégies de recherche à savoir : déterministe, heuristique, stochastique. Et/ou ayant des comportements différents (évolutionnaire, essaim, local…etc,).

3.3.2 SFS et SBS

SFS (**S**equential **F**orward **S**election) ou (sélection séquentielle croissante) est une heuristique proposée pour la sélection de caractéristiques (Marill et Green [1963]). SFS démarre par un sous-ensemble vide et à chaque itération, la meilleure caractéristique sera ajoutée au sous-ensemble sélectionné et supprimé de l'ensemble de départ. Cette opération est répétée jusqu'à un critère d'arrêt. Contrairement à l'algorithme SFS, la sélection séquentielle arrière SBS (Whitney [1971]) commence par l'ensemble entier des caractéristiques et à chaque itération, la caractéristique la plus mauvaise sera supprimée (Algorithme 2.2). une comparaison de ces

deux méthodes a été faite par Ahan et Bankert (Aha et Bankert [1995]). Les résultats de l'étude montre l'avance de la méthode SBS vu qu'elle calcule le score de l'interaction d'une caractéristique avec un ensemble de caractéristiques plus large, relativement au SFS qui ne prend en considération que l'interaction de cette caractéristique avec le sous-ensemble actuel. En revanche, la méthode SBS est très coûteuse en termes de temps car elle évalue des sous-ensembles de grande taille.

3.3.3 La méthode Tabou

L'algorithme Tabou Search (TS) est une méta-heuristique proposée au début par Golver (Glover, 1986). TS est une procédure de recherche heuristique locale qui tente d'explorer l'espace de solutions au-delà d'une optimalité locale. L'un des principaux composants de TS est l'utilisation de la mémoire adaptative. L'algorithme TS utilise une procédure de recherche de voisinage pour passer d'une solution potentielle *x* à une solution améliorée *x'* dans le voisinage de *x*. Ceci est effectué de façon itérative, jusqu'à ce qu'un critère d'arrêt défini soit satisfait. La stratégie de déplacement d'une solution à une autre qui est voisine est toujours sous la contrainte d'éviter que les dernières solutions soient récemment visitées. Cette contrainte est garantie par la construction d'une liste appelée, Tabu List en anglais (TL). TS peut accepter les mouvements en descente (downhill) pour éviter d'être piégé dans les maxima locaux (critères d'aspiration). Les stratégies à court et à long terme sont utilisées pour atteindre respectivement l'intensification et la diversification, et elles sont utilisées pour améliorer la recherche. La stratégie d'intensification se concentre sur les régions prometteuses du domaine réalisable. D'un autre côté, la diversification tente de trouver une nouvelle solution dans une large zone de l'espace de recherche. (H. Zhang & Sun, 2002), (Hedar, Wang, & Fukushima, 2008), ont appliqué la méthode Tabou pour la sélection d'attributs et pour la réduction d'attributs dans la théorie des ensembles approximatifs respectivement. La figure 3.2 montre la procédure qui décrit un code simple de TS.

1- Select an initial solution x.
2- Generate Neighbors solutions x' from x,
3- Accept the best solution from x' if its movement doesn't exist in the Tabu List.
3- Add the movement that has generated the x' solution to the Tabu List TL.
4- If a chosen number of iteration has elapsed then stop,
Otherwise set x=x' and return to 2.

Figure 3-2 : pseudo code de l'algorithme Tabou

3.3.4 Recuit simulé

L'algorithme de recuit simulé (Kirkpatrick, 1984), est une méthode d'optimisation probabiliste qui émule le processus physique de chauffage d'un matériau, en se basant sur l'abaissement lent de la température pour diminuer les défauts, et minimisant ainsi l'énergie du système. Dans chaque itération, et à partir d'une solution courante on génère une solution voisine en modifiant l'état actuel, selon un critère prédéfini. La nouvelle solution est acceptée si elle améliore la fonction objective. Pour le cas contraire, cette dernière sera aussi acceptée mais avec une probabilité. La probabilité d'acceptation dépend quasiment de la température actuelle et le numéro d'itération. Ce critère est définit par Metropolis, (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953). Sachant bien que la température de recuit diminue dans une plage $[t_{debut}, t_{fin}]$ suivant un taux de refroidissement prédéfini. De cette façon la méthode évite l'optimum local en acceptant des solutions médiocres au début de recherche. En revanche et à la fin de recherche ces solutions seront évidemment rejetées. En conséquent et avec cette stratégie, l'algorithme de Recuit-simulé converge.

```
Algorithm (SA)
    1- Initialiser les paramètres ;
    2- S= solution initiale ;
    3- T=T0 ;  // température initiale
    4- while  i < Max Itération
       Générer S'  à partir de S
            Si F(s')   < F(s)
                 S=S'
                    Sinon
                       Δ= F(s') - F(s)
                        R=random() ;
                        Si (r<exp(-Δ /k * T )
                                S=S'
       T= a* T ;
       Inc(i) ;
    5- End_while
    6- End Algorithm
```

Figure 3-3 : pseudo code de la méthode Recuit simulé

3.3.5 Les Algorithmes génétiques (GA)

Les algorithmes génétiques (GAs) sont une sorte d'algorithmes évolutifs, basés sur l'inspiration naturelle (Holland & Reitman, 1978). Ils imitent les opérations de mutation, de croisement et de sélection des gènes dans la nature. Les GAs se basent sur le principe de combinaison entre deux solutions courantes (Parents) afin d'en produire deux nouvelles. En faisons cette reproduction, les GAs tentent d'améliorer le score de la fonction objective. La mutation de son côté permet de perturber ou de modifier une partie d'une solution existante afin d'éviter de tomber dans une solution optimale prématurée. Quant à la sélection, le troisième opérateur de GAs, la sélection permet de choisir les parents qui vont participer à la production des générations suivantes. GAs avaient plusieurs domaines d'applications et notamment dans celui de la sélection des caractéristiques. Par exemple, les GA ont été utilisés

comme un outil pour résoudre le problème de sélection de caractéristiques dans le contexte de l'exploration de données (Siedlecki et Sklansky, 1989).

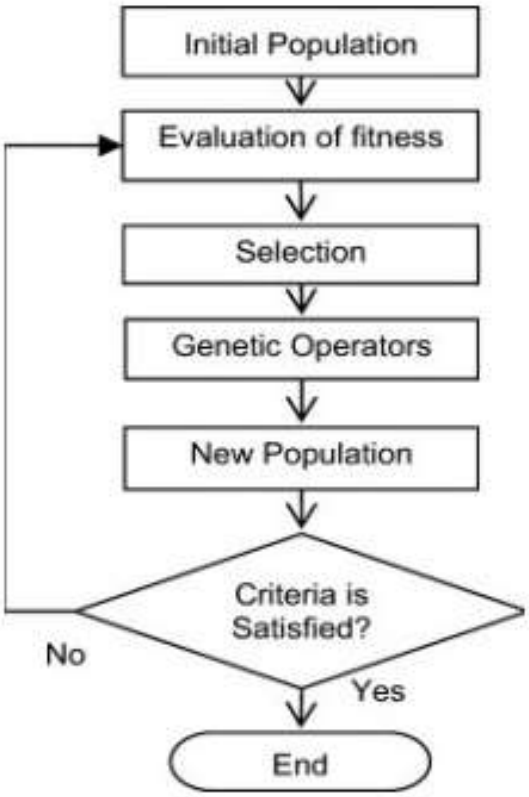


Figure 3-4 : Schéma général d'algorithmes génétiques

3.3.6 Les colonies de fourmis (ANT)

Les colonies de fourmis, Ant Colony Optimization (ACO) en anglais, est une approche méta heuristique appartenant à la famille des méthodes d'essaims intelligentes. ACO est une méthode de recherche aléatoire basée sur le comportement alimentaire des essaims (fourmis) (DORIGO, 1992). Plus précisément, chaque fourmi dépose la même quantité de phéromone pendant qu'elle suit un certain chemin. Cette quantité de phéromone attire les autres fourmis à suivre ce même chemin. La probabilité de choix d'un chemin donné parmi d'autres voies est proportionnelle à la quantité de phéromone présente dans cette voie : plus cette dernière dispose d'une quantité de phéromone importante plus elle est favorisée, l'inverse est vrai. Simultanément, la quantité de phéromone présente dans les différents trajets diminue en fonction d'un taux d'évaporation prédéfini. Par conséquent, les trajets courts gardent plus de phéromone au fil du temps et ils seront plus susceptibles d'être sélectionnés

par les fourmis. Dans (Aghdam, Ghasem-Aghaee et Basiri, 2009), les auteurs ont appliqué une méthode de FS basée sur l'algorithme ACO pour la catégorisation de texte. En outre, (Al-Ani, 2005) propose une méthode de FS avec l'algorithme BCA qui est appliqué au problème de la classification de la parole. Les figures 3.5 à 3.7 montrent le comportement des fourmies sur une expérience durant une période de temps.

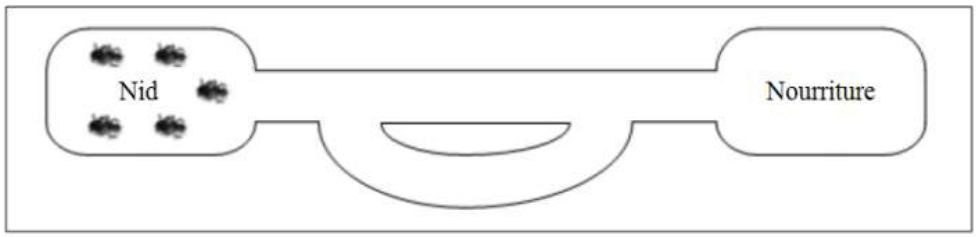


Figure 3-5 L'état initial du système.

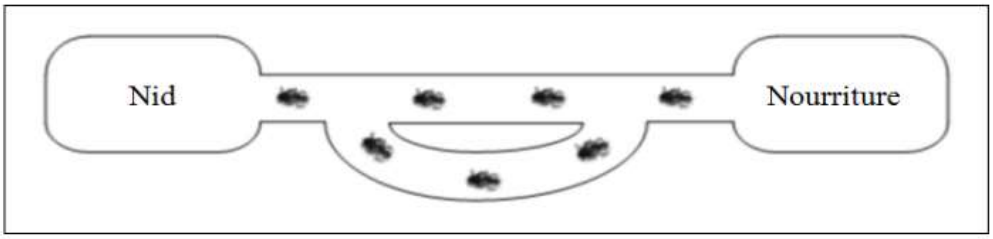


Figure 3-6 Comportement des Fourmis après au début de simulation.

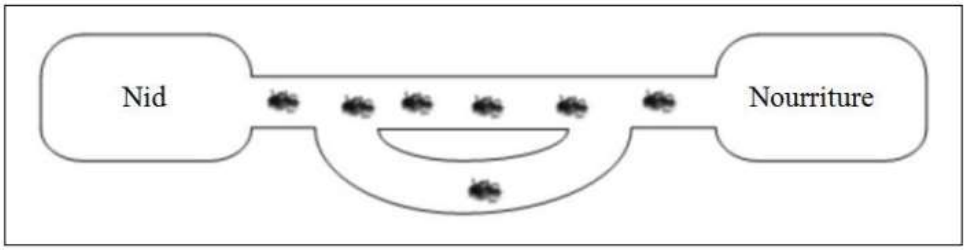


Figure 3-7 Comportement des Fourmis après une période de temps.

3.3.7 L'optimisation par essaims particulaires (PSO)

L'optimisation par essaims particulaires, Particle swarm optimization (PSO), inventée par Kennedy et Eberhart (Kennedy & Eberhart, 1995) s'inspire du comportement des animaux et du comportement social humain. Initialement, cette approche a été développée pour des problèmes continus. PSO est une recherche basée sur la population, qui vise à trouver une

solution sous-optimale dans l'espace de recherche. Chaque individu (particule) dans l'essaim et pendant la recherche, change itérativement sa nouvelle position basée sur sa vitesse, sa meilleure position locale trouvée jusqu'à présent et la meilleure position globale. PSO a ensuite été adaptée pour une optimisation discrète (Kennedy & Eberhart, 1997), qui prend en charge la sélection des attributs et utilise des états binaires comme 0 et 1.

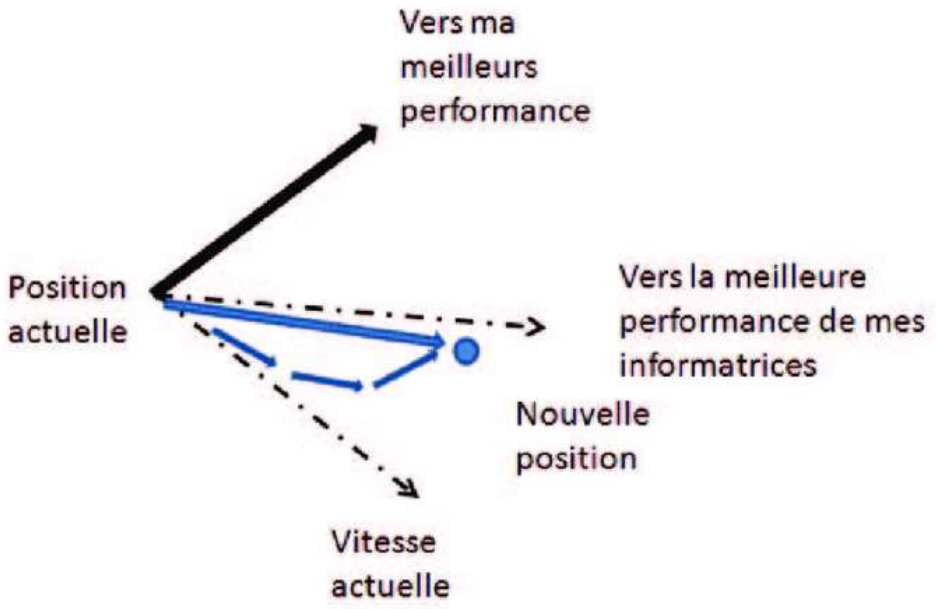


Figure 3-8 schémas de principe du déplacement d'une particule.

3.3.8 L'optimisation des essaims particulaires géométrique (GPSO)

L'optimisation des essaims de particules géométriques est l'une des versions améliorées de PSO (Shi & Eberhart, 1998), appelée Geometric Particle Swarm Optimization (GPSO). Elle étend la recherche à d'autres espaces de recherche comme les espaces Euclidien, Manhattan et Hamming. Dans le GPSO, la particule se déplace vers sa nouvelle position à l'aide d'une opération de croisement multifonctionnelle à masque, avec la restriction qui respecte quatre exigences pour être une combinaison convexe dans un certain espace. En particulier, dans l'espace de Hamming, un croisement à trois parenté (3PMBCX) a été proposé dans (Moraglio, Di Chio, & Poli, 2007). Le nouveau mouvement des particules est introduit comme suit: "*Given three parents a, b and c in {0, 1} n, generate randomly a crossover mask of length n with symbols from the alphabet {a, b, c}. Build the offspring filling each element with the bit from the parent appearing in the crossover mask at the position*".

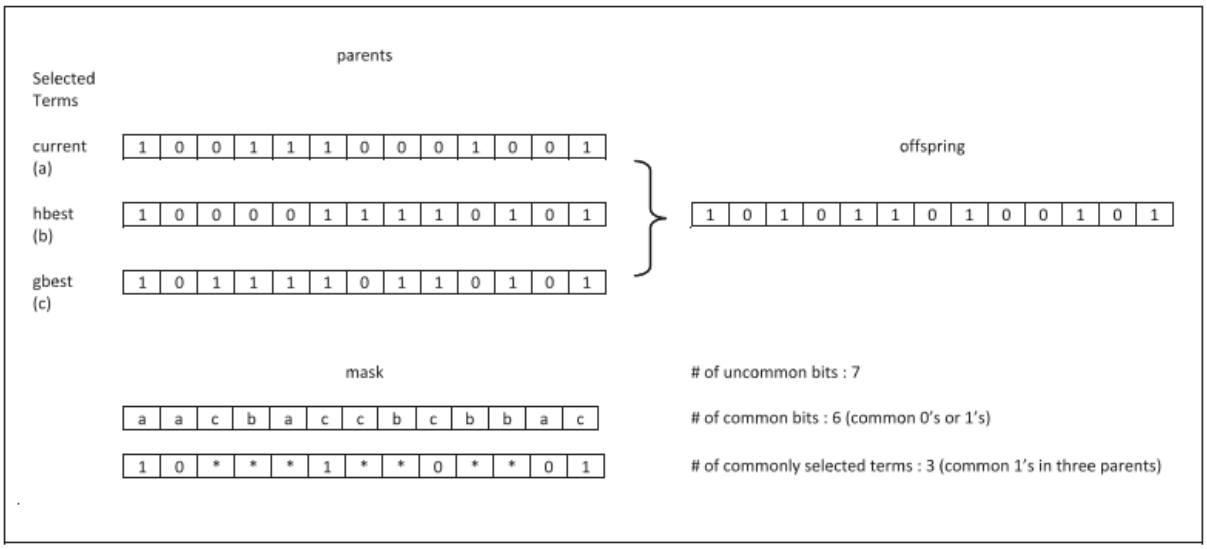


Figure 3-9 sélection d'un terme à partir de 3 parents avec le masque 3pBMCX (Karabulut, 2013)

Dans la Figure 3-9, nous présentons un pseudo code de l'algorithme GPSO dans l'espace de Hamming. Pour une particule donnée *i*, le masque de croisement (3PMBCX) est composé de trois parents : la position actuelle X_i, la meilleure position historique trouvée h_i, et g la meilleure position globale. Les poids w_1 , w_2 et w_3 indiquent la probabilité de prendre la valeur de chaque parent: x_i, h_i et g, respectivement dans le masque de croisement.

```
1- Initialize positions of the swarm.
2- WHILE stop criteria not satisfied DO
3- FOR all particle i DO
4- IF fitness (xi) is better than fitness (hi) THEN
5- hi=xi
6- IF fitness (hi) is better than g THEN
7- g = hi
8- END FOR
9- FOR all particle i DO
10- Xi+1 =3PMBX(<xi,w1>,<hi,w2>,<g ,w3>)
11- mutate (xi+1)
12- END FOR
13- END WILE
```

Figure 3-10 pseudo code de l'algorithme GPSO

3.3.9 Algorithme de papillons de nuit (MFO).

L'algorithme d'optimisation de papillons de nuit, ou, Moth-Flame optimization (MFO), proposé par Mirjalili, (Mirjalili, 2015) est un algorithme d'essaims particulaires inspiré des papillons de nuit. Selon l'auteur, l'idée derrière peut se présenter comme suit : le fait le plus intéressant au sujet des papillons de nuit est leurs méthodes de navigation spéciales dans la nuit. Ils ont été développés pour voler pendant la nuit en utilisant la lumière de la lune. Ils utilisent un mécanisme appelé orientation transversale pour la navigation. En effet, il a été observé que les papillons volent en spirale autour des lumières proches, et également suivent une trajectoire droite pour celles qui est distantes. La figure suivante montre un schéma conceptuel de leur comportement.

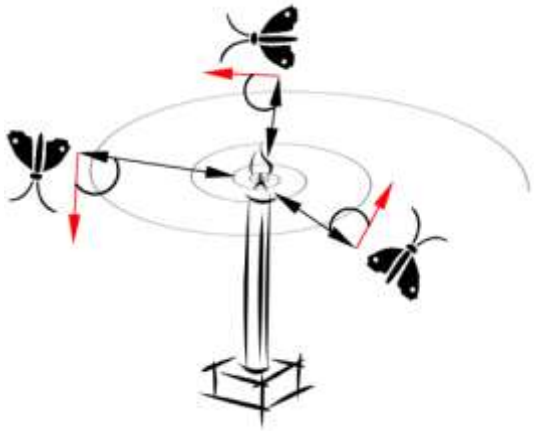


Figure 3-11 mouvement spirale des papillons de nuit sur la source de lumiere.

Un comportement artificiel cependant opté par l'auteur, étant le mécanisme de mise à jour de papillons de nuit. Ce dernier est exprimé par une spirale logarithmique pour cette méthode. Particulièrement, La formule utilisée dans l'algorithme MFO est définit comme suit :

$$\boldsymbol{s(m_i, f_i) = D_i . e^{bt} . \cos(2\pi t) + F_j} \qquad 3.1$$

Dont D_i est la distance entre le $i^{ème}$ papillon du $j^{ème}$ papillon, donnée par : $\|F_i - M_i\|$.

Le comportement des papillons de nuit artificiels dans l'algorithme MFO est définit comme suit : après l'initialisation des papillons et le calcul du score, chaque papillon *i* désigne un autre papillon *j*, le papillon *j* sera la source lumineuse du papillon *i*. Cela est fait selon un mécanisme de sélection. Ensuite, la modification des positions pour chacun des papillons est faite selon la formule spirale 3.1. Les opérations de sélection et déplacement seront répétées

jusqu'à l'atteinte d'un seuil donné. Pour plus de détails nous recommandons la lecture de , (Mirjalili, 2015) .

3.3.10 L'algorithme de Levy Flight (Levy_SCA)

En raison que l'algorithme Sinus Cosinus (SCA) soufre du problème des minima locaux en raison de la convergence prématurée dans des problèmes complexes d'optimisation non linéaire, ce nouvel algorithme basé sur le Levy Flight a été proposé et il présente également une variante de SCA. Dans l'algorithme Levy_SCA, (N. Li, Li, & Deng, 2017), le comportement d'un agent lui permet de tester sa valeur actuelle de la fonction objective avec ses valeurs historiques afin de marquer si l'individu risque de tomber ou pas dans les minima locaux. Dans ce sens, les individus marqués subissent une mise à jour de leurs positions différentes en utilisant les paramètres de la fonction Levy Flight. Ce qui améliore la capacité de recherche globale de l'algorithme dans la phase d'exploration et la capacité de recherche locale dans la phase d'exploitation. Le mécanisme Levy-Flight est une stratégie de marche aléatoire, qui appartient au mouvement de Brownian généralisé, et sa longueur de pas du mouvement de distance suit une distribution aléatoire non-Gaussienne, qui a appelé aussi la distribution stable Levy. Mathématiquement, une version simple de la distribution de Levy peut être définie comme suit :

$$\begin{cases} \sqrt{\frac{y}{2\prod}}\ \exp\left[-\frac{y}{2(s-u)}\right]\frac{1}{(s-u)^{\frac{3}{2}}}\ ,(0<u<s<inf) \\ \\ 0,(s\leq 0) \end{cases} \quad (3.2)$$

Où le paramètre *u* est le paramètre de localisation et le paramètre *y (y> 0)* est un paramètre scalaire qui contrôle la distribution d'échelle.

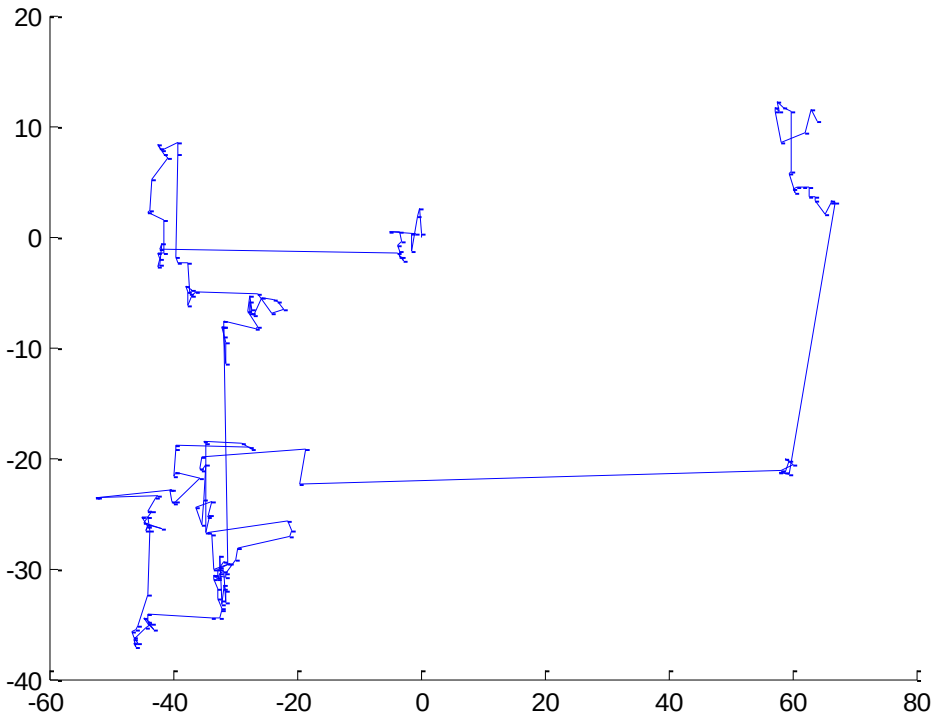


Figure 3-12 illustrationde la fonction Levy Flight

D'un autre côté, l'algorithme SCA que nous allons détailler plus tard dans le chapitre 04, et comme tous les autres algorithmes d'optimisation globale, comporte les deux stratégies essentielles de l'intensification et de la diversification. Pour la diversification, l'algorithme doit avoir un opérateur aléatoire afin d'explorer aléatoirement mais aussi globalement l'espace de recherche. Hors que la phase d'exploitation se réfère à la capacité de recherche locale autour des régions prometteuses obtenues dans la phase d'exploration. L'algorithme SCA s'appuie sur les deux fonctions mathématiques Sinus et Cosinus durant son processus de recherche. La figure ci-dessous montre les stratégies de l'intensification et de la diversification de SCA.

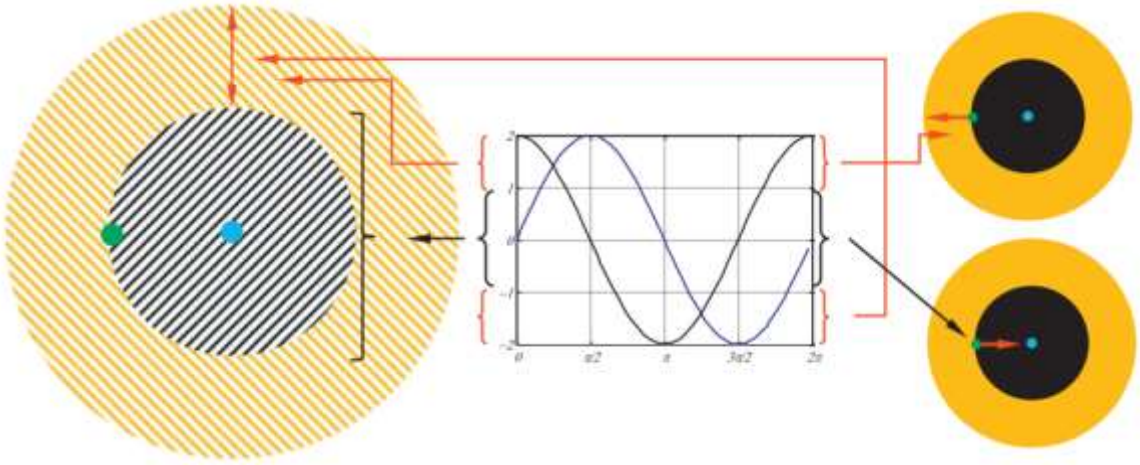


Figure 3-13 les stratégies de l'intensification et de la diversification de l'algorithme SCA.

3.3.11 L'algorithme SCA pondéré (Weighted SCA)

L'algorithme *Weighted SCA* est proposé dans le cadre d'améliorer l'ancien algorithme SCA afin d'atteindre une meilleur exploration dans l'espace de recherche. Dans cette méthode, en plus de la position actuelle d'un agent et ainsi son score par rapport à la fonction objective, on attribue à chaque agent un poids en fonction de son classement. La position de chaque agent de recherche est mise à jour en fonction de la position moyenne de certains autres agents, de la meilleure solution obtenue et de la position précédente pondérée de la position de l'agent en cours de modification. Plus en détails, les poids des agents sont affectés dans un ordre décroissant selon leur score de la fonction objective, le meilleur score aura le plus grand poids et vice versa. La formule 3.3 montre la façon de le calculer.

$$\boldsymbol{w}_{x_j^t} = \mathbf{1} - \boldsymbol{j} \,.\, (\frac{\mathbf{1}}{\boldsymbol{population\ size}}) \qquad \textbf{(3.3)}$$

Où *j* représente le rang de l'agent dans la population. Ce rang est définit selon son score. Par ailleurs, le nombre d'agents participant et qui seront concernés par le calcul de la moyenne est définit selon la formule 3.4. Ce nombre est itérativement dégradé afin d'avoir un équilibre dans l'exploration et l'exploitation.

$$c_t = \textbf{Population size} - t \cdot \left(\frac{1}{\textbf{Max itteration}}\right) \quad (3.4)$$

La valeur de la moyenne des positions pondérées est calculée à partir du terme C_t, cette dernière sera un terme important pour la formule finale de déplacement des agents par laquelle on en aura plus d'exploration. Voir la formule.

$$\mathbf{Avg_t} = \frac{\mathbf{sum}\left[\mathbf{w_j^t x_j^t(1)}, \mathbf{w_j^t x_j^t(2)}, \ldots\ldots \mathbf{w_j^t\, x_j^t(c)}\right]}{\mathbf{c_t}} \quad (3.5)$$

A partir de de la moyenne des positions et les positions pondérées de chacun des agents, nous arrivons à la formule finale et qui présente la mise-à-jour des positions de la population. La formule montre les détails.

$$\boldsymbol{X(j+1)_{t+1}} = \begin{cases} \boldsymbol{Avg + r1 * sin(r2) * \left|r3 * P(j) - w_j^t x_j^t\right|} & \boldsymbol{if\ r4 < 0.5} \\ \\ \boldsymbol{Avg + r1 * cos(r2) * \left|r3 * P(j) - w_j^t x_j^t\right|} & \boldsymbol{if\ r4 \geq 0.5} \end{cases} \quad (3.6)$$

Où *P(j)* est la position de la meilleure solution. Les grandes valeurs du paramètre C_t au début du processus d'optimisation conduisent à : chaque agent va mettre à jour sa position sous l'influence d'un certain nombre d'agents de recherche, par la valeur (Avg) résultante de leur moyenne et ce qui donne une bonne exploration. D'un autre côté, en abaissant la valeur de (C_t), la position d'un agent sera de plus en plus dépendante de la position de la meilleure solution. Par conséquent, on aura une concentration de recherche sur l'exploitation.

3.4 Conclusion

Dans ce chapitre nous avons exposé les algorithmes de recherche qui représentent le noyau du processus des méthodes de sélection. La qualité du sous ensemble sélectionné dépend fortement de la performance d'algorithmes de recherche. Ces derniers peuvent se classer selon la stratégie de leur recherche ainsi que sur leurs comportements et inspirations. En résumé, nous avons introduit les différentes stratégies de recherche dans lesquelles nous avons mis l'accent sur les algorithmes utilisés dans la sélection des attributs et notamment les algorithmes de recherche stochastiques. Dans cette perspective, nous avons présenté les

algorithmes de recherche les plus reconnus dans la littérature, comme les GAs, PSO et Aco, et notamment un ensemble particulier d'algorithmes de recherche très récents. Ces derniers sont inspirés de l'algorithme SCA qui présente la base de notre inspiration et contribution. Dans le chapitre suivant nous allons présenter la partie contribution de cette thèse qui est un nouvel algorithme de recherche baptisé ISCA (Improved Sin Cosine Algorithm) pour la sélection de termes.

Chapitre 04

Contribution :

un nouvel algorithme de recherche

pour la sélection des termes.

4.1 Introduction

Dans un processus de classification textuelle, la phase de sélection des attributs (FS) est une étape importante dans le prétraitement des données afin de remédier au problème de haute dimensionnalité, connu aussi par le nom de malédiction. Elle vise à déterminer un sous-ensemble restreint contenant un nombre non seulement limité de caractéristiques pertinentes mais aussi suffisant. Ceci est dans le but de maintenir ou améliorer les performances des algorithmes de classification.

Ce chapitre est organisé comme suit : nous commençons par une présentation du processus de classification dans lequel on montre notre stratégie de sélection, la représentation des sous-ensembles des termes ainsi que leur évaluation. L'algorithme SCA original est ensuite présenté en détail dans la section 4.6 accompagné par une discussion et quelques critiques qui motivent notre proposition. Dans cette même section, nous décrivons également notre nouvel l'algorithme de recherche ISCA proposé pour la sélection des termes. Nous expliquons notre stratégie de diversification ainsi que la stratégie de génération des nouvelles solutions. Enfin, la section 4.7 conclut ce chapitre.

4.2 Notre processus de classification de textes

Le processus de classification de textes TC est généralement divisé en quatre phases : la phase de prétraitement et représentation du texte, la phase de réduction de la dimensionnalité par sélection de termes, la tâche de classification, et enfin la phase d'évaluation des résultats. Dans notre travail, nous suivons la stratégie de classification textuelle présentée dans la figure 4.1. Dans ce schéma, nous utilisons deux étapes de sélection pour la phase FS : une étape de type filtrage et une autre étape de type Wrraper pour la sélection. A noter que, dans la première étape (filtre), les tops 200 attributs qui sont obtenus en s'appuyant sur la mesure statistique Gain d'Information (IG) sont retenus. Ceux-ci représentent les données d'entrée (Input) pour l'étape suivante où l'algorithme ISCA est utilisé comme étant une procédure de recherche. Avec cette stratégie à deux étapes, le nombre d'attributs (200) rend l'utilisation de notre algorithme ISCA plus pratique et efficace. Finalement, dans l'étape d'évaluation des attributs, nous appliquons l'algorithme Naïve Bayes qui est aussi utilisé pour la classification finale.

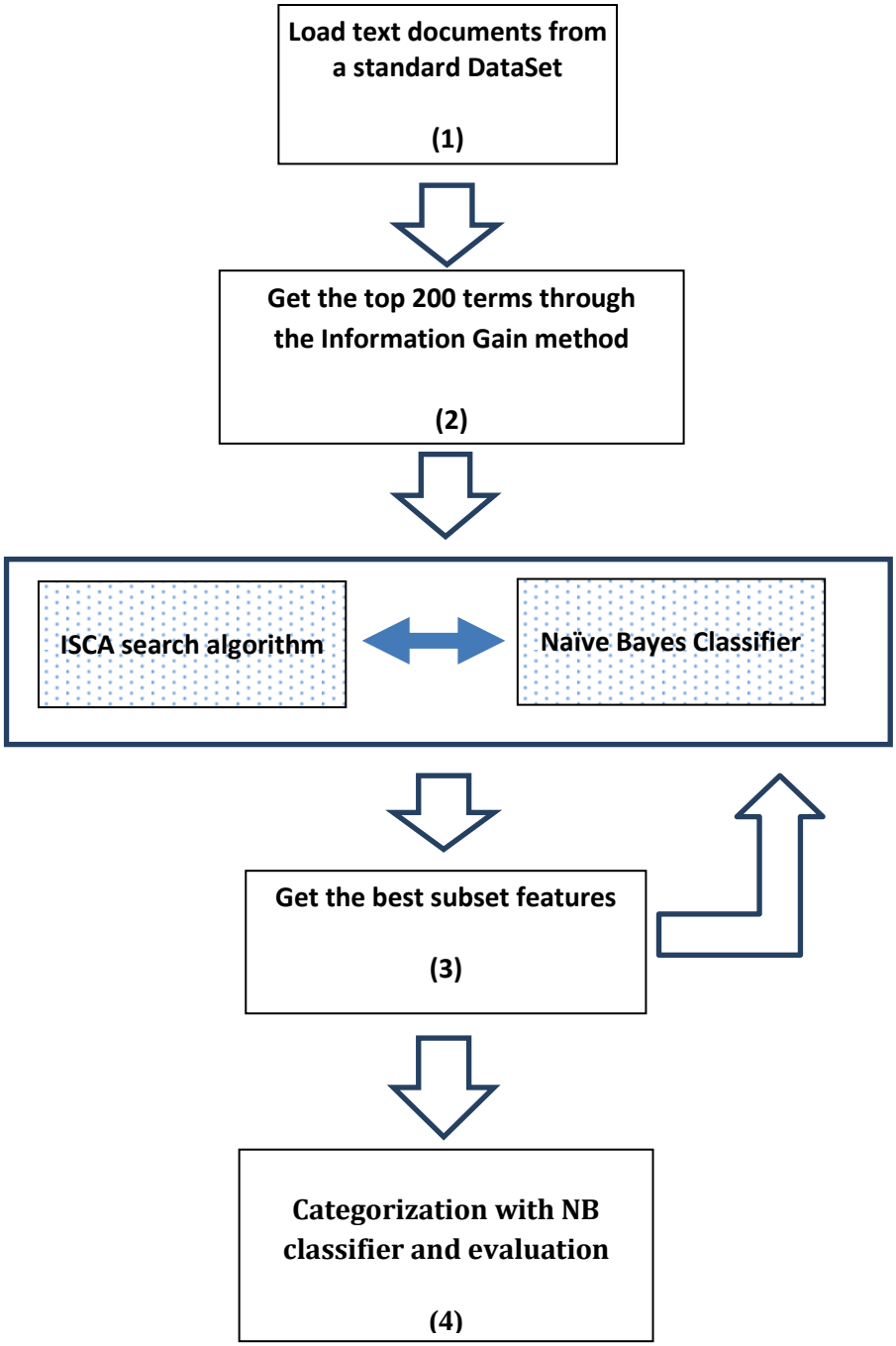


Figure 4-1 Notre processus de classification de textes.

4.3 Représentation des sous-ensembles d'attributs

Une solution potentielle (un sous-ensemble d'attributs) est représentée par un vecteur d'attributs où chaque attribut (terme) correspond à une dimension et chaque variable se fait attribuée une valeur réelle dans l'intervalle [0, 1]. La décision de sélectionner ou de rejeter un attribut est prise comme suit: si la valeur de la position est supérieure ou égale à *0.5*, la caractéristique correspondante est sélectionnée; sinon, cette fonctionnalité est rejetée (Lin,

Hsu, & Hung, 2012). Dans le calcul du score d'individus, un seuil est utilisé pour déterminer le sous-ensemble d'attributs à évaluer, comme indiqué par l'équation 4.1 suivante :

$$f_{i,j} = \begin{cases} \mathbf{1} & \boldsymbol{if\ \ Xi,j \geq 0.5} \\ \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases} \qquad 4.1$$

Dans cette équation, $X_{i,j}$ est la valeur de dimension d'agent de recherche *i* dans la dimension *j*. Rappelons que lors de la mise à jour de la position des agents dans certaines dimensions, la valeur mise à jour peut dépasser les limites de l'intervalle [0, 1]. Pour surmonter ce problème, nous avons utilisé une simple règle de troncature pour garantir le non-dépassement des bornes limites des variables.

4.4 Evaluation du sous-ensemble de caractéristiques

L'évaluation des attributs est un processus itératif appliqué sur chaque génération. La priorité d'un sous-ensemble de caractéristiques est donnée à la base des valeurs de scores calculées en utilisant une fonction objectif qui consiste, dans notre algorithme ISCA, à maximiser la précision et de tenter d'atteindre un sous-ensemble restreint. Pour ce faire, nous avons combiné le nombre de caractéristiques et le taux d'erreur en une seule formule. Par conséquent, l'objectif global est de minimiser la fonction objectif de l'équation 4.2 ci-dessous. Notez que cette même formule est utilisée pour les autres algorithmes de recherche qui ont été testés dans ce travail.

$$\boldsymbol{fit = \alpha * Error\_Rate + (1 - \alpha) * \left(\frac{number\ of features\ selected}{total\ features}\right)} \qquad \mathbf{4.2}$$

où *fit* est la fonction objectif et α est une constante réelle entre *0* et *1*. Le taux d'erreur est pondéré par α, tandis que $(1 - \alpha)$ représente l'importance relative du nombre d'attributs sélectionnés. Dans la fonction d'évaluation que nous avons appliquée dans la méthode Wrapper, le classificateur Naïve Bayes (NB) (McCallum & Nigam, 1998) est utilisé dans le but d'évaluer la qualité d'un sous-ensemble de candidat de termes.

Afin d'évaluer les performances globales de notre algorithme Wrapper, plusieurs expériences sont effectuées sur plusieurs jeux de données textuelles. L'indicateur de performance (*F1 – mesure)* du classificateur Naïve Bayes est utilisé pour calculer le taux d'erreur impliqué

dans l'équation 4.2. De plus, nous avons appliqué une méthode de test à base d'une validation croisée 3-folds sur un jeu de données d'apprentissage. Par le biais de méthodes empiriques, α est fixé à 0,9 qui a donné de bons résultats.

4.5 Un nouvel algorithme de recherche ISCA

4.5.1 Introduction

Notre contribution consiste en la proposition d'un algorithme de recherche pour la sélection des termes, capable d'explorer le mieux possible l'espace de recherche dans un temps raisonnable. Notre algorithme ISCA, *Improved Sin Cosin Algorithm*, est combiné avec la mesure statistique IG pour cerner le problème de haute dimensionnalité. Cette hybridation a permis de réduire le coût d'exécution en termes de temps et de ressources, et par conséquent d'offrir une application plus pratique de notre algorithme. De plus, la stratégie d'exploration proposée dans cet algorithme lui permet d'atteindre des résultats très satisfaisants.

En effet, notre algorithme est une version améliorée de l'algorithme original SCA qui a été choisi pour les raisons suivantes : en se référant à la littérature, SCA et ses variantes ont montré leur efficacité dans plusieurs domaines. On démontre qu'ils sont mieux placés par rapport à des algorithmes d'optimisation présents dans l'état de l'art, y compris l'algorithme PSO, les GAs et l'ACO dans de nombreuses situations. Ce succès nous a motivé à axer nos travaux sur la proposition d'une nouvelle variante de SCA dont l'objectif primordial est de booster en plus les résultats de performance de la phase de FS dans un contexte de catégorisation de texte. L'efficacité de notre méthode hybride de FS ainsi que notre méthode Wrapper qui s'appuie sur notre nouvel algorithme ISCA sont démontrées par des expérimentations significatives qui seront décrites dans le chapitre suivant.

Dans ce qui suit, nous décrivons les principes de base de l'algorithme SCA. Nous montrons aussi ses limites et ses inconvénients. La méthode de représentation et le codage de données sont introduits avec le critère d'évaluation. Dans la section 4.5.4. nous décrivons notre nouvel algorithme en expliquant l'idée principale de son comportement et en détaillant ses stratégies de diversification et intensification. Enfin, nous concluons avec un résumé de nos contributions exposées dans ce chapitre.

4.5.2 Présentation et critique de l'algorithme SCA

Dans cette section, nous présentons les fondements de l'algorithme SCA dont notre inspiration est fondée. Le comportement de cet algorithme est décrit afin de monter ses

stratégies d'intensification et de diversification, les formules qui y sont relatives ainsi que son pseudocode. Nous enchaînons par une critique de cet algorithme, ce qui va éclaircir nos motivations pour introduire notre nouvel algorithme ISCA.

4.5.3 L'algorithme (SCA)

L'algorithme Sinus - Cosinus (SCA) est un algorithme stochastique itératif, relativement récent, proposé dans (Mirjalili, 2016). A chaque itération, les positions des solutions sont mises à jour sur la base de la fonction sinus ou cosinus comme indiqué par les équations 4.3 et 4.4 respectivement:

$$\boldsymbol{X(i,j)_{t+1} = X(i,j)_t + r1 * \sin(r2) * |r3\, P(j)_t - X(i,j)_t|} \qquad 4.3$$

$$\mathbf{X(i,j)_{t+1} = X(i,j)_t + r1 * \cos(r2) * |r3\, P(j)_t - X(i,j)_t|} \qquad 4.4$$

Où $X(i,j)_t$ est la position de la solution courante i en j^{eme} dimension à la t^{eme} itération, et la position du meilleur individu en j^{eme} dimension à l'itération t. Le signe | | indique la valeur absolue et r_1, r_2 et r_3 sont des variables qui prennent des valeurs aléatoires.

Les deux équations ci-dessus peuvent être combinées dans l'équation suivante:

$$\boldsymbol{X(i,j)_{t+1} = \begin{cases} X(i,j)_t + r1 * sin(r2) * |r3 * P(j)_t - X(i,j)_t| & if\ r4 < 0.5 \\ X(i,j)_t + r1 * cos(r2) * |r3 * P(j)_t - X(i,j)_t| & if\ r4 \geq 0.5 \end{cases}} \qquad 4.5$$

Sachant qu'ici r_4 est une variable réelle aléatoire entre 0 et 1. La signification des paramètres r_1,r_2, r_3 et r_4 est intuitivement donnée comme suit: le paramètre r_1 indique la région à côté des positions (ou la direction du mouvement) qui peut être soit dans l'espace situé entre la solution actuelle et la destination, soit en dehors de cet espace . Le paramètre r_2 définit dans quelle sens le mouvement va se produire : vers la destination ou dans la direction inverse. Le paramètre r_3 représente un poids aléatoire à la distance entre la solution actuelle et la destination. Enfin, le paramètre r_2 figure comme argument (qui indique l'angle) dans les

fonctions sinus et cosinus de l'équation 4.5. Le pseudocode de SCA est donné dans l'algorithme 1. Ci-dessous. Pour obtenir un équilibre entre l'exploitation et l'exploration, (Mirjalili, 2016) utilise un changement adaptatif dans le domaine du sinus et du cosinus dans les formules 4.3 et 4.4. précédentes . La stratégie optée pour assurer cet équilibre est exprimée mathématiquement dans la formule 4.6 suivante :

$$\boldsymbol{r1 = a - a * \frac{t}{T}} \qquad 4.6$$

Où *t* est l'itération courante, *T* est le nombre maximum d'itérations et *a* est une constante.

Algorithme 1 : Algorithme d'optimisation globale Sinus Cosinus.

1. Initialiser un ensemble d'agents aléatoires
2. Calculer la fonction de coût de chaque agent
3. Sélectionner le meilleur agent **Best** qui optimise la fonction de coût
4. Mettre à jourr_1,r_2, r_3 et r_4
5. Mise à jour de la position des agents de recherche à l'aide de l'équation (4.5)
6. Si (t <nombre maximum d'itérations (T)), allez à 2.
7. Renvoyer la meilleure solution **Best** comme étant un optimum global

Avant de dévoiler la nouvelle stratégie de diversification de notre algorithme ISCA, continuons toujours sur l'algorithme SCA mais cette fois ci, on va présenter quelques limites et critiques de celui-ci. Malgré son succès dans plusieurs applications d'optimisation, l'algorithme SCA, comme d'ailleurs d'autres algorithmes méta-heuristiques, n'est pas parfait et présente certains inconvénients qui doivent être surmontés. En effet, SCA a l'aptitude d'être bloqué dans des régions sous-optimales à cause de l'effort de calcul requis pour obtenir de meilleures performances. La raison de cela peut résider dans une limite dans son processus d'exploration de l'espace de recherche, car nous pouvons observer la concentration de la mise à jour des solutions vers ou à partir de la destination (meilleure solution

déjà trouvée). Par conséquent, si on rend cet algorithme capable de se référer à d'autres solutions provenant d'autres régions de l'espace de recherche, nous aurons plus de chance de découvrir des régions prometteuses et donc d'éviter la convergence vers un optimum local.

4.5.4 Le nouvel algorithme de recherche ISCA

La contribution principale de notre travail est la proposition d'un nouvel algorithme de recherche de type Wrapper, appelé Algorithme Amélioré de Sinus Cosinus (ISCA). En effet, ISCA est une version améliorée du récent algorithme *Sine Cosine Algorithm* (SCA) proposé par Mirjalili (Mirjalili, 2016). L'algorithme ISCA tente de découvrir encore plus de nouvelles régions dans l'espace de recherche relativement à l'original. Contrairement à SCA qui se concentre uniquement sur la meilleure solution pour en générer une nouvelle, le nouvel algorithme (ISCA) tient en compte deux positions de solutions : la première est la position de la meilleure solution trouvée jusqu'à présent, et une seconde qui est une position aléatoire prise dans l'espace de recherche. Cette combinaison va conduire à la fois à éviter la convergence prématurée et à améliorer les performances de recherche. Notre idée intuitive est de rendre la génération de la solution potentielle dépendante non seulement de la meilleure solution, mais aussi d'une autre position de solution aléatoirement générée. La figure 4.3 montre la différence dans l'espace découvert entre les deux algorithmes SCA et ISCA. Sur cette base, il est évident de constater à quel point la zone de l'ISCA est plus large que la zone originale du SCA. Pour concrétiser cette idée, nous suggérons que la nouvelle position d'une solution (i) soit régie par la formule suivante :

$$\boldsymbol{X(i,j)_{t+1} = (1 - r1) * X(i,j)_t + r1 * R} \qquad 4.7$$

Où $X(i,j)_t$ est la position de la solution courante i dans la j^{eme} dimension de la t^{eme} itération. R est une position aléatoire comprise dans l'intervalle de l'espace de recherche. r_1 est la même variable aléatoire utilisée dans la formule 4.6 relative à l'algorithme SCA. La figure 4.3 expose une simulation dans laquelle la valeur de r_1 est itérativement rabaissée. Cette dernière affecte directement les valeurs de la fonction sinus utilisées dans notre formule de déplacement d'agents (cf. Le système d'équation 4.8).

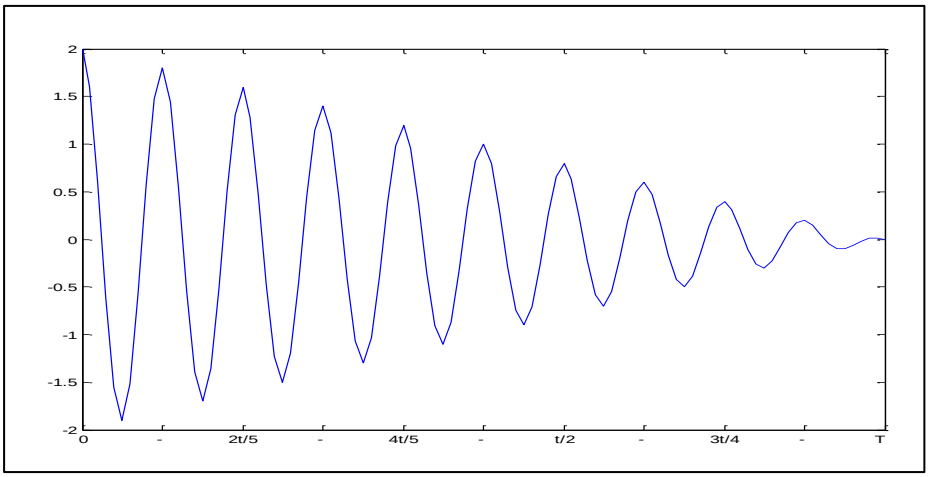


Figure 4-3 l'évolution de la fonction Sinus suivant le paramètre r1, (a=2).

Les poids $(1 - r1)$ et $(r1)$ représentent les poids relatifs à la position actuelle $X(i,j)_t$ et de la position aléatoire *R* respectivement. Ces paramètres affectent la position de la nouvelle solution. En effet, cette dernière est influencée par la position aléatoire *R* au début de la recherche. Cette influence est dégradée itérativement par le premier paramètre. Par conséquent, à la fin de la recherche, la position de la nouvelle solution tend vers sa position actuelle. Cette transaction d'attraction entre la position *R* et la position actuelle exprime le passage de la divergence à la convergence.

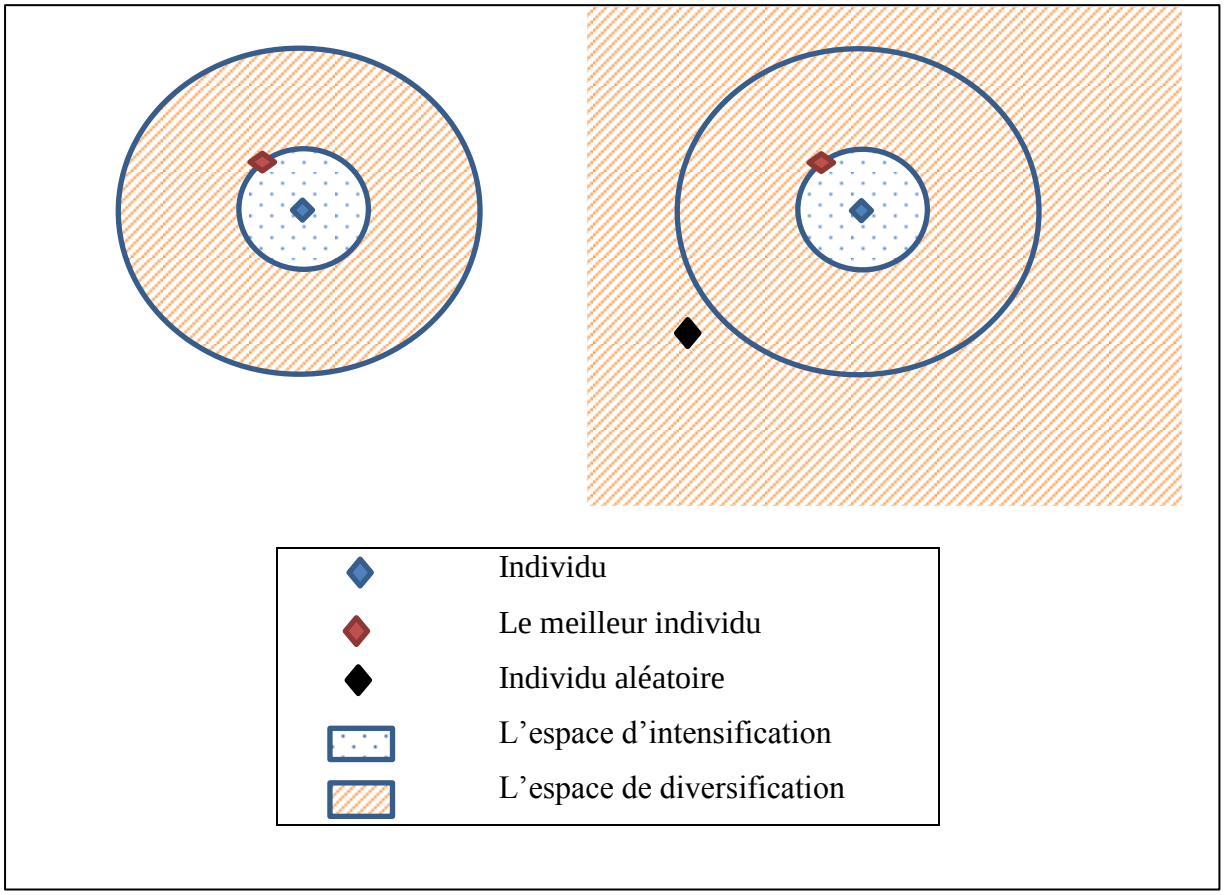


Figure 4-4 exploration et exploitation des algorithmes SCA et ISCA, l'algorithme SCA en haut;et l'algorithme Sin Cosine amélioré en bas

De plus, nous apportons quelques modifications au système d'équations 4.5). D'une part, nous proposons de fusionner les deux formules sinus et cosinus en une seule formule: au lieu de basculer entre deux formules, nous suggérons de limiter la mise à jour de position par une seule formule parmi les deux précédentes. En effet, nous pensons que le choix entre les deux formules est arbitraire parce que les deux fonctions sinus ou cosinus renvoient des valeurs entre -1 et +1 pour une variable entrée comprise dans l'intervalle $[0, 2\pi]$. Par conséquent, nous pensons que l'exemption d'alternance entre ces deux fonctions n'infligera pas des changements sur les performances de l'algorithme ISCA. Par ailleurs, pour atteindre un bon compromis entre l'intensification et la diversification, nous introduisons une nouvelle équation combinée par les deux variables aléatoire c et r_1. Elle assure une transition efficace entre exploration et exploitation. L'équation 4.8 ci-dessous montre les mises à jour des nouvelles positions d'agents ainsi que la stratégie de diversification:

$$X(i,j)_{t+1} = \begin{cases} (1 - r1) * X(i,j)_t + r1 * R & if\ c < r1 \\ P(j)_t + r1 * \sin(r2) * |r3 * P(j)_t - X(i,j)_t| & if\ c \geq r1 \end{cases} \quad 4.8$$

Sachant que 'C' dans l'équation ci-dessus est donné par $c = b * r_4$. Le paramètre b est un entier généralement supérieur à 1 qui sert à amplifier le paramètre c. Plus ce dernier est grand plus on aura de chance d'avoir de la diversification, sinon on intensifie la recherche sur la meilleure solution P. Les étapes de sélection des caractéristiques de l'approche proposée sont données dans Algorithme 4.2. Notons que nous avons mis l'accent sur l'amélioration de la diversification de l'algorithme original SCA tout en maintenant sa force d'intensification. Dans cette optique, nous avons modifié le système d'équations pour mettre à jour la position de solutions d'une façon adéquate et adaptative. En utilisant cette modification, ISCA atteint de bonnes performances au sujet de sélection de caractéristiques pour la catégorisation de textes. A noter que ceci est réalisé sans utiliser un grand nombre de paramètres de configuration, comme c'est le cas dans l'algorithme SCA.

L'algorithme 4.2 ci-dessous illustre le pseudocode de notre algorithme ISCA. Lors de la première étape, les positions des solutions sont initialisées de façon aléatoire dans l'espace de recherche. Ensuite, une boucle de ($max\_iteration$) étapes est exécutée pour ressortir un sous-ensemble optimal de caractéristiques. À chaque itération de cette boucle, le score (Fit) de chaque solution est calculé à l'aide de la fonction objectif définie dans la formule 4.2. Après cette phase d'évaluation, la meilleure solution ($Best\_pos$) de la population est déterminée en fonction du meilleur score. La mise à jour des positions des solutions est réalisée en utilisant le système d'équations 4.8 dans lequel chaque agent balance entre ses deux formules, servant respectivement aux stratégies de diversification et d'intensification. Cette permutation est contrôlée à l'aide des deux paramètres $r1$ et $r4$, où $r1$ décroît régulièrement à chaque itération tandis que la variable $r4$ reçoit des valeurs aléatoires comprises entre 0 et 1. On note ici que la diversification est effectuée si l'inégalité ($r4 < r1_1$) est vérifiée, sinon l'intensification est effectuée. De plus, la mise à jour des paramètres ($r1$,$r4$) est ajustée afin d'obtenir une transaction appropriée de la stratégie de diversification vers l'intensification, ce qui garantit un bon équilibre entre elles.

Algorithme 4-2: ISCA (nb_agent,dimension_size, max_iteration,a,b)

Initialiser les positions des agents (X).

% (Fit) est le cout calculé par la fonction objectif d'un agent donné

Calculer (Fit) pour tout agent, et sélectionner la meilleure position (Best_pos).

T ←2;

Tanque *(t <= max_iteration)*

$r_1 \leftarrow a - t*(a/ max\_iteration);$ % décroitre r1 itérativement

Pour Chaque agent *(X)*, dans la dimensionnalité *(j)*

$r_2 \leftarrow (2*pi)*rand();$

$r_3 \leftarrow rand();$

$r_4 \leftarrow b*rand ();$

% modification de position de la solution

si $(r_4 < r_1)$

% obtenir une position aléatoire dans l'espace de recherche *[lb,ub]*

m=Get_rand_position();

$X(j)=(1-r_1)* X(j)+(r_1)*m;$

Sinon

$X(j)= Best\_pos (j)+r_1*(sin(r_2))*abs(r_3* Best\_pos (j)-X(j));$

Fin-si

% fin de mise à jour de la position

Fin-pour

Recalculer le (*Fit*) pour chaque agent (*X*), et obtenir la meilleure position (Best_pos).

Inc (*t*);

Fin-tanque;

FIN-ALGORITHME

Afin de valider notre nouvel algorithme ISCA, une série d'expériences va être effectuées sur des collections de textes très reconnues, à savoir : Reuters, Ohsumed et 20NG. Notre algorithme sera aussi comparé avec plusieurs algorithmes de recherche qui sont aussi très reconnus dans la littérature. En outre, une seconde comparaison est accompagnée avec des algorithmes de même famille que le nôtre, dérivés d'algorithme SCA pour en donner plus de

crédibilité. Une discussion à ce sujet sera faite au chapitre suivant, à la suite de la présentation des résultats de nos expérimentations.

4.6 Conclusion

Dans ce chapitre, nous avons proposé un nouvel algorithme de recherche ISCA pour la sélection des attributs à partir des documents textuels. L'algorithme ISCA est une version améliorée de l'algorithme original SCA (*Sine Cosine Algorithm*). Il offre une meilleure exploration de l'espace de recherche afin d'améliorer les résultats et notamment la performance de la classification. Nous avons montré que cette capacité d'exploration est garantie grâce à la stratégie suivante: on tient en compte deux positions de solutions, la première position est toujours celle de la meilleure solution déjà trouvée, tandis que la deuxième est une autre position aléatoire sélectionnée à partir de l'espace de recherche. Avec cette combinaison, nous estimons obtenir une meilleure exploration de ce dernier, ce qui conduit à sélectionner le meilleur sous-ensemble de termes.

Le chapitre suivant est dédié à la validation de l'algorithme en procédant à des expérimentations et des discussions afin de mettre en relief les améliorations apportées par notre algorithme relatives à la qualité du classificateur.

Chapitre 05

Réalisation et validation de note contribution

5.1 **Introduction**

L'algorithme ISCA proposé a été évalué sur un grand nombre d'expériences. On outre, pour évaluer et valider notre proposition nous avons réalisé une comparaison entre les algorithmes de recherche les plus reconnus dans le domaine de FS dont certains sont déjà appliqués au problème TC, notamment GA, ACO, SCA , OBL-SCA Levy_Sca, MFO et Weighted_Sca. Dans ce but, nous avons utilisé différents jeux de données, collections, des extraits de Benchmarks très reconnus dans l'application TC, particulièrement : Reuters21-578, TREC et Ohsumed. Notons que le code source des algorithmes de FS comparés a été implémenté dans l'environnement Matlab2 013 dans lequel l'algorithme Naive Bayse (NB) a été utilisé dans la tâche de catégorisation. Par ailleurs, on a réalisé des expériences sur une machine équipée d'un processeur 2,40 GHz, possédant 8 cœurs et 08 Go de RAM. Il convient de mentionner ici que la plupart des résultats de performance sont indiquées à la fois qualitativement et quantitativement.

5.2 **Collections de Textes**

La performance des algorithmes de TC est évaluée à l'aide de jeux de données, les plus couramment utilisés dans la communauté de fouille de texte (Rossi, Marcacini, & Rezende, 2013). Dans cette étude, nous avons sélectionné neuf collections textuelles, extraites à partir des trois bases de données textuelles très reconnus, notamment Reuters 21-578, Ohsumed et TREC. Ces collections de textes sont décrites ci-dessous en termes de leurs étiquettes de classe et le nombre de documents par classe.

5.2.1.1 Collection Re0

La collection RE0 (Forman, 2006) est composée d'articles de Reuters-21578 collection (D. D. Lewis, 2004). Le tableau 5.1 présente le nombre de documents pour chaque classe de la collection Re0.

5.2.1.2 Collection La1s

La collection La1s (Forman, 2006) est composée d'articles de presse du Los Angeles Times, extraits de (TREC, 2013). Le tableau 5.2 présente le nombre de documents pour chaque classe de la collection La1s.

Tableau 5-1 Nombre de documents par classe pour la collection Re0.

ordre	étiquettes de classe	Abs # Docs
1	Lei	11
2	Wpi	15
3	logement	16
4	vente au détail	20
5	Ipi	37
6	Bop	38
7	Emplois	39
8	Réserves	42
9	Cpi	60
10	Gnp	80
11	l'intérêt	219
12	Commerce	319
13	Argent	608
Nombre total de documents		1504

Tableau 5-2 Nombre de documents par classe de la collection La1s.

Ordre	Etiquette de classe	Abs # Docs
1	Nationale	28
2	Étranger	301
3	Divertissement	375
4	Financier	487
5	Des sports	759
6	Métro	905
Nombre total de documents		3075

5.2.1.3 Collection La2s

La collection de La2s (G. Forman, 2006) est composée d'articles de presse du Los Angeles Times extraits de (TREC, 2013). Le tableau 5.3 présente le nombre de documents pour chaque classe de la collection La2s.

Tableau 5-3 Nombre de documents par classe de la collection La2s.

ordre	étiquette de classe	Abs # Docs
1	nationale	273
2	Étranger	341
3	Divertissement	354
4	Financier	555
5	Des sports	738
6	Métro	943
Nombre total de documents		3204

Tableau 5-4 Nombre de documents par classe de la collection Oh0.

ordre	Étiquettes de classe	Abs # Docs
1	Mexique	51
2	Acide urique	56
3	6-Ketoprostaglandin-F1-alpha	57
4	Larynx	66
5	Chimie du cerveau	71
6	Créatine –Kinase	76
7	Éthique	115
8	Fundus-Oculi	136
9	Angleterre	181
10	Prothèse valvulaire	194
Nombre total de documents		1003

5.2.1.4 Collection Oh0

La collection Oh0 (G. Forman, 2006) fait partie de la collection OHSUMED (Hersh et al., 1994). Le tableau 5.4 présente le nombre de documents pour chaque classe de la collection Oh0.

5.2.1.5 Collection Oh5

La collection Oh5 (G. Forman, 2006) fait partie de la collection OHSUMED (Hersh et al., 1994). Le tableau 5.5 présente le nombre de documents pour chaque classe de la collection Oh5.

Tableau 5-5 Nombre de documents par classe de la collection Oh5.

ordre	étiquettes de classe	Abs # Docs
1	Solutions	59
2	Développement de l'enfant	61
3	Radiation-Dosage	61
4	Audiométrie	72
5	Les anticoagulants	74
6	Microsomes	85
7	Azote	93
8	Cou	120
9	Phospholipides	144
10	Greffe-survie	149
Nombre total de documents		918

5.2.1.6 Oh10 collection

La collection Oh10 (G. Forman, 2006) fait partie de la collection OHSUMED (Hersh et al., 1994) . Le tableau 5.6 présente le nombre de documents pour chaque classe de la collection Oh10.

Tableau 5-6 Nombre de documents par classe de la collection Oh10.

ordre	étiquettes de classe	Abs # Docs
1	Etudes d'échantillonnage	52
2	Italie	60
3	Maladies rétiniennes	61
4	Nutrition	70
5	Livraison	87
6	Les spermatozoïdes	116
7	Acides Arachidoniques	126
8	Hématocrite	148
9	Analyse des coûts et des coûts	165
10	En mangeant	165
Nombre total de documents		1050

5.2.1.7 Collection Oh15

La collection Oh15 (G. Forman, 2006) fait partie de la collection OHSUMED (Hersh et al., 1994) . Le tableau 5.7 présente le nombre de documents pour chaque classe de la collection Oh15.

Tableau 5-7 Nombre de documents par classe de la collection Oh15.

ordre	étiquettes de classe	Abs # Docs
1	Aluminium	53
2	Adénosine-diphosphate	56
3	Leucine	56
4	Urémie	66
5	Facteurs de coagulation sanguine	69
6	Vaisseaux sanguins	98
7	Mémoire	98
8	Mouvement cellulaire	106
9	Activation Enzymatique	154
10	Infections à staphylocoques	157
Nombre total de documents		913

5.2.1.8 Collection FBIS

La collection FBIS (Foreign Broadcast Information Service) (G. Forman, 2006) fait partie de la collection TREC (TREC, 2013). Cette collection est composée d'articles de journaux du monde entier. Le tableau 5.8 présente le nombre de documents pour chaque classe de la collection FBIS.

5.2.1.9 Collection Tr41 :

La collection Tr41 (G. Forman, 2006) est dérivée de la collection (TREC, 2013). Les classes correspondent aux documents jugés pertinents pour des requêtes particulières. Le tableau 5.9 présente le nombre de documents pour chaque classe de la collection Tr41.

Tableau 5-8 Nombre de documents par classe pour la collection FBIS.

ordre	étiquettes de classe	Abs # Docs
1	95	38
2	4	43
3	11	46
4	119	46
5	240	46
6	3	48
7	161	65
8	100	92
9	108	94
10	221	119
Nombre total de documents		637

Tableau 5-9 Nombre de documents par classe pour la collection Trec41.

ordre	étiquettes de classe	Abs # Docs
1	356	9
2	359	18
3	351	26
4	358	33
5	355	35
6	360	83
sept	353	95
8	357	162
9	352	174
Dix	354	243
Nombre total de documents		878

5.3 **Méthodologie d'évaluation**

Les mesures de précision et de rappel sont couramment utilisées pour l'évaluation des algorithmes de catégorisation. La Précision (P) désigne le rapport du nombre de documents correctement attribués dans la catégorie *C* au nombre total de documents classés comme appartenant à la catégorie *C*. Le Rappel (R) présente le rapport du nombre de documents correctement attribués dans la catégorie *C* au nombre total de documents appartenant réellement à la catégorie *C*. De plus, une troisième mesure commune est la mesure F (FM) qui

désigne la moyenne harmonique de précision et de rappel. Les trois mesures sont décrites dans les équations suivantes (5.1) à (5.3):

$$P = TP / (TP + FN). \quad 5\text{-}1$$

$$R = TP / (TP + FP). \quad 5\text{-}2$$

$$F1 = 2PR / (P + R). \quad 5\text{-}3$$

Dans lesquelles, *TP* est le nombre de vrais-positifs, *FP* représente le nombre de faux positifs et *FN* représente le nombre de faux-négatifs.

Quand il s'agit de la catégorisation multi-classes, la moyenne Micro et la moyenne Macro des indicateurs : précision et rappel sont généralement utilisées. La moyenne Macro pèse de manière égale toutes les classes, peut import le nombre de documents qu'elles contiennent. La moyenne Micro pondère également tous les documents, favorisant ainsi la performance sur des classes fréquentes. De plus, le micro *F1* dépend en grande partie de catégories fréquentes tandis que la macro *F1* est influencée par chaque catégorie. Afin de comparer les performances sur toutes les catégories et chaque catégorie, les deux Micro *F1* et Macro *F1* sont utilisées dans ce travail. Eqs. (1.3) - (1.5) montrent le calcul de Micro-moyenne et la Macro-moyenne de la précision, le rappel et F-mesure pour |c| indépendants problèmes de classification binaire (silva & B Ribeiro, 2010).

$$\boldsymbol{Micro\ P} = \frac{\sum_{i=1}^{c} \boldsymbol{TPi}}{\sum_{i=1}^{c} (\boldsymbol{TPi} + \boldsymbol{FNi})} \quad 5\text{-}4$$

$$\boldsymbol{Maco\ P} = \frac{\sum_{i=1}^{c} \boldsymbol{Pi}}{|\boldsymbol{c}|} \quad 5\text{-}5$$

$$\boldsymbol{Micro\ R} = \frac{\sum_{i=1}^{c} \boldsymbol{TPi}}{\sum_{i=1}^{c} (\boldsymbol{TPi} + \boldsymbol{FPi})} \quad 5\text{-}6$$

$$\boldsymbol{Maco\ R} = \frac{\sum_{i=1}^{c} \boldsymbol{Ri}}{|\boldsymbol{c}|} \qquad 5\text{-}7$$

$$\boldsymbol{Micro\ F1} = \frac{\boldsymbol{2 * micro\ P * micro\ R}}{\boldsymbol{micro\ P + micro\ R}} \qquad 5\text{-}8$$

$$\boldsymbol{Maco\ F1} = \frac{\sum_{i=1}^{c} \boldsymbol{F1i}}{|\boldsymbol{c}|} \qquad 5\text{-}9$$

5.4 Résultats et discussion

Nous avons fait plusieurs expériences sur différents jeux de données. De plus, pour mettre en évidence la crédibilité de notre algorithme, celui-ci a été testé et comparé avec de nombreux autres algorithmes. Ces algorithmes peuvent se répartir en deux types : interne et externe. Le premier type comporte l'algorithme original SCA ainsi que trois de ses versions améliorées, notamment Levy_SCA, Weighted_SAC et OBL_SCA. Ensuite, nous avons pris à partir de la littérature trois algorithmes de recherche très reconnus, particulièrement les GAs, Aco et l'algorithme MFO. Ces derniers présentent le deuxième type d'algorithmes de notre comparaison.

Afin de valider notre algorithme ISCA proposé, plusieurs paramètres ont été testés dans la phase expérimentale, et les meilleures valeurs de ces paramètres sont retenus. Dans cette étude, nous suivrons la démarche expérimentale utilisée dans (Aghdam et al., 2009) afin de montrer l'efficacité du nouvel algorithme ISCA dans la sélection des attributs.

5.4.1 Résultats et Comparaison interne

Dans cette partie, nous avons comparé nos résultats avec ceux d'autres algorithmes de recherche, notamment l'algorithme original SCA, ainsi certains de ses successeurs, à savoir: OBL-SCA, Levy SCA et Weighted _SCA afin d'offrir plus de fiabilité à nos expériences. En ce qui concerne les paramètres d'algorithmes, nous assignons les valeurs 30 et 50 aux paramètres : nombre de population et nombre maximal d'itérations respectivement. Les autres valeurs des paramètres sont présentées suivant chaque algorithme dans le tableau 5.10.

Tableau 5-10 les paramètres de SCA, OBL-SCA, Levy_SCA , Weighted_SCA , et de l'algorithme ISCA proposé.

Ordre	Algorithme de recherche	Paramètres
01	SCA	a = 2
02	OBL_SCA	a = 2
03	Levy_SCA	a = 2; α = 1,5
04	Weighted_SCA	a = 2
08	ISCA	a = 1; b = 8

Le tableau 5.11 présente la précision, le rappel et F1-measure du classificateur NB combiné avec les sous-ensembles d'attributs sélectionnés par les algorithmes SCA, OBL -SCA, Levy-SCA, Weighted-SCA et de l'algorithme ISCA proposé. Plus exactement, nous avons présenté par ligne les moyennes des valeurs de précision, du rappel et de la mesure F1 pour toutes les catégories des collections textuelles mentionnées ci-dessus.

Tableau 5-11 Moyennes de précision (P), de rappel (R) et de F-mesure (F1) de SCA, OBL-SCA, Levy_SCA et l'ISCA proposé les 9 collections de texte

Colle-ction	Wgh –ISCA			Levy-SCA			Obl_SCA			ISCA		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Re0	65,09	63,28	62,66	72,49	72,04	70,78	65,89	67,37	65,02	68,67	71	69
La1s	70,29	72,44	70,64	73,35	75,84	73,91	72,14	74,87	72,6	76,13	76,88	76,1
La2s	72,98	74,43	72,95	76,59	77,03	76,4	72,98	75,87	73,33	78,23	79,22	78,18
Oh0	70,21	79,39	72,45	77,07	85,36	79,37	72,18	81,09	74,2	78,85	87,19	81,52
Oh5	71,05	75,82	72,08	80,09	86,16	81,88	72,38	78,8	73,83	81,59	86,82	83,42
Oh10	65,36	70,82	66,6	70,95	76,98	72,42	66,02	72,29	67,34	73,05	78,41	74,48
Oh15	66,89	74,85	68,87	73,76	80,45	75,62	68,56	75,57	70,2	76,67	83,92	79,02
Fbis	74,9	73,05	73,2	80,58	79,24	79,3	75,26	73,97	73,9	81,39	80,01	80,2
Trec41	83,34	80,83	81,08	86,79	84,67	85,17	83,35	82,03	81,95	88,23	86,49	86,9
AVG	71,12	73,88	71,17	76,85	79,75	77,20	72,08	75,76	72,49	78,09	81,10	80,98

D'après les résultats du tableau 5-11, Il est évident de constater que notre algorithme ISCA est nettement meilleur en matière de valeurs Macro-Moyenne des métriques : précision, rappel et F1-measure, et cela dans la plupart des collections. Sachant que ces 03 algorithmes de recherche, avec lesquelles nous avons comparé notre algorithme ISCA, sont très compétitifs. Les résultats obtenus montrent clairement l'avance de notre algorithme ISCA. On conclusion, on peut dire que l'algorithme ISCA est très utile en matière de sélection de termes pour la catégorisation de textes.

Tableau 5-12 Micro-F1 et Macro-F1 de SCA, OBL-SCA, Levy_SCA , Weighted_SCA , MFO, ACO, GA et de l'ISCA proposé sur les 9 collections de textes (en pourcentage) .

Collection		Algorithme de recherche			
		WG-SCA	Levy-SCA	Obl-SCA	ISCA
Re0	macrof1	62,66	70,78	65,02	73,97
	microf1	67,68	72,68	69,31	75,46
La1s	macrof1	70,64	73,91	72,6	76,1
	microf1	74,76	77,55	76,18	79,02
La2s	macrof1	72,95	76,4	73,33	78,18
	microf1	76,96	79,67	77,52	81,59
Oh0	macrof1	72,45	79,37	74,2	81,52
	microf1	74,5	80,27	76,07	81,62
Oh5	macrof1	73,09	83,36	75,52	83,89
	microf1	72,08	81,88	73,83	83,42
Oh10	macrof1	66,6	72,42	67,34	74,48
	microf1	70,38	76,01	70,68	77,57
Oh15	macrof1	68,86	75,62	70,2	79,02
	microf1	71,38	77,56	72,55	80,39
Fbis	macrof1	73,2	79,3	73,9	80,2
	microf1	80,25	85,12	80,55	85,83
Trec41	macrof1	81,08	85,17	81,95	86,9
	microf1	87,12	90,03	87,81	91,14

Dans le tableau 5.12, nous présentons les résultats de calcul des valeurs de la moyenne exprimés en micro-F1 et macro-F1. Pour les quatre algorithmes utilisés dans cette expérimentation, les résultats indiquent clairement que notre algorithme ISCA proposé dépasse considérablement tous les autres algorithmes, que ce soit en matière de Macro-F1 ou également en matière de Micro-F1. Ce constat est remarqué en général dans la plupart des jeux de données textuelles considérés et sans exception. Néanmoins, on remarque une légère compétitivité avec l'algorithme Levy SCA.

Dans la partie suivante nous présentons une comparaison qualitative des expériences précédentes à travers des figures. À ce titre, les figures 5.1 et 5.2 montrent graphiquement la variation de performances des algorithmes de recherche FS suivant le pourcentage de caractéristiques sélectionnées. Particulièrement, cette figure représente une synthèse globale sur les 9 collections de textes en exprimant les moyennes de la mesure F1 en termes de micro-moyenne et macro-moyenne. Autrement dit, la valeur exprimée représente la moyenne des moyennes obtenues dans les 9 collections. L'axe vertical affiche la moyenne de la mesure F1 (micro/macro) tandis que l'axe horizontal représente le pourcentage du nombre de caractéristiques prises. Il est clair que notre algorithme proposé ISCA a obtenu les meilleures performances en comparaison avec les autres algorithmes utilisés dans cette partie.

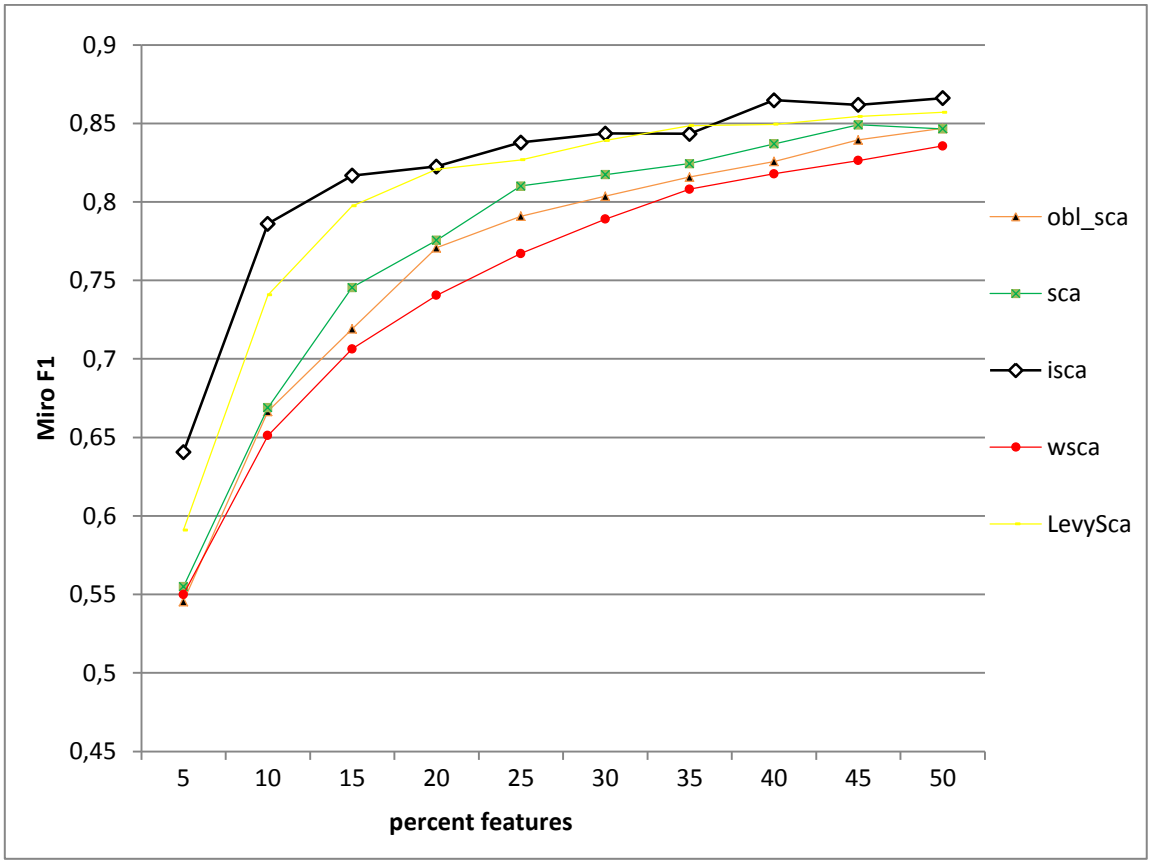


Figure 5-1 Moyennes de Micro-F1 de SCA, OBL-SCA, Levy SCA , Weighted SCA , MFO, ACO, GA et de l'ISCA proposé pour la sélection d'attributs sur les 9 collections en augmentant le nombre de termes.

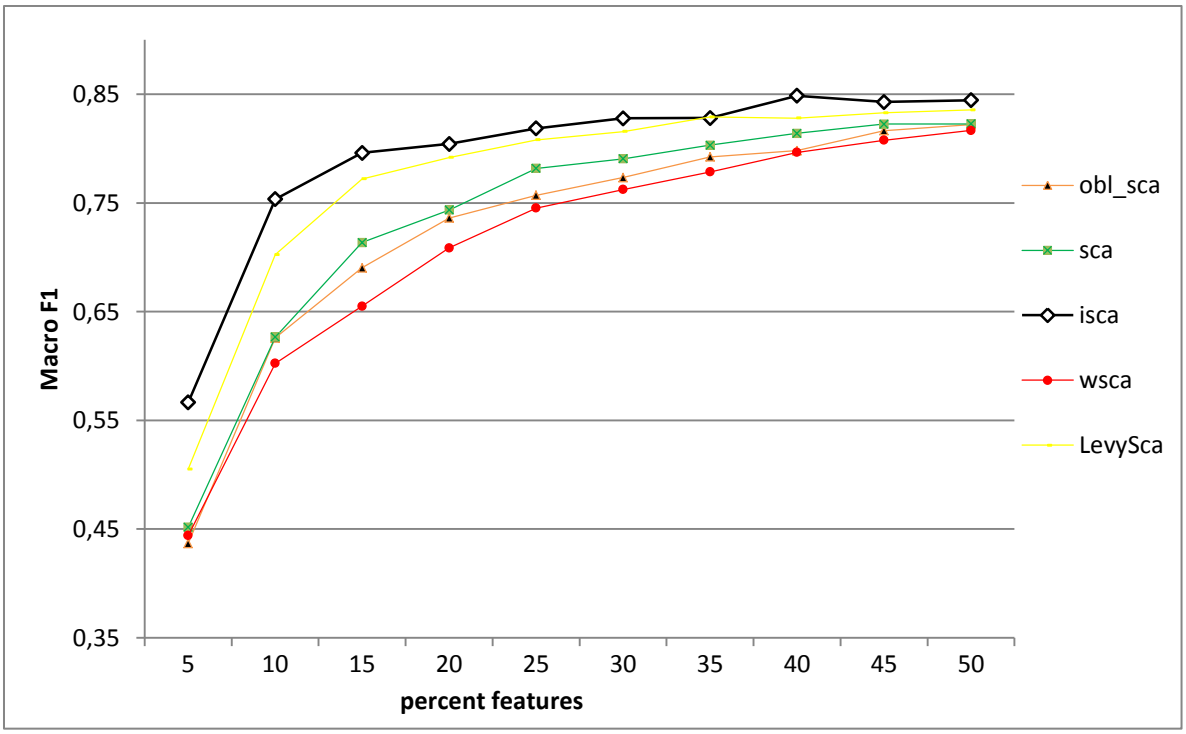


Figure 5-2 Moyennes de Micro F1 de SCA, OBL-SCA, Levy SCA , Weighted SCA , MFO, ACO, GA et de l'ISCA proposé pour la sélection d'attributs sur les 9 collections en augmentant le nombre de termes.

5.4.2 Comparaison Externe

Dans la partie précédente nous avons montré l'avance de notre algorithme par rapport aux autres algorithmes comme WG-SCA et Levy-Flight, etc. Pour avoir plus de crédibilité, nous avons effectué une deuxième expérience similaire à la première. Mais cette fois-ci, nous faisons appel à d'autres algorithmes de recherche en dehors de la famille SCA. La particularité de ces algorithmes est qu'ils ont déjà été utilisés dans des travaux antérieurs portant sur la sélection dattributs et ont fait preuve d'une performance remarqable. La comparaison introduite comporte, l'algorithme de recherche MFO qui est un algorithme d'optimisation, *Moth-Flam Optimization* (MFO), les algorithmes génétiques (GA) et l'optimisation par colonies de fourmies (Ant colony optimisation, ACO). Pour tous les algorithmes mentionnés ci-dessus, nous assignons les valeurs 30 et 50 aux paramètres : Nombre de Population et nombre maximal d'itérations respectivement. Les autres valeurs des paramètres de chaque algorithme sont présentées dans le tableau 5-13.

Tableau 5-13 SCA, OBL-SCA, Levy_SCA , Weighted_SCA , MFO, ACO , GA et les paramètres de paramètres ISCA proposés

Rang	Algorithme de recherche	Paramètres
05	MFO	b = 1
06	ACO	Phéromone initiale = 1
		$\alpha = 1$; $\beta = 0,1$; $\sigma = 0,2$
07	GA	Probabilité de croisement = 0,7
		Probabilité de mutation = 0,05
08	ISCA	a = 1; b = 8

Le tableau 5.14 présente la précision, le rappel et la mesure F1 du classificateur NB combiné avec les sous-ensembles d'attributs sélectionnés par les algorithmes MFO, ACO et les GAs et également avec notre algorithme ISCA. En particulier, les lignes présentent les moyennes de précision, les moyennes du rappel et de F1-mesure de toutes les catégories des collections mentionnées. D'après les résultats ci-dessus, il est facile de constater que notre algorithme ISCA est nettement meilleur en termes de valeurs de Macro Moyenne de : P, R et F1 dans la plupart des collections. Malgré que ces 3 algorithmes de recherche testés sont fiables, notre algorithme arrive à les dépasser. Pour conclure, les résultats obtenus montrent clairement

l'avance de notre algorithme ISCA et qu'il est très bénéfique à la sélection des caractéristiques textuelles.

Tableau 5-14 moyenne s de précision (P), le rappel (R) et F-mesure (F1) de SCA, OBL-SCA, Levy_SCA et l'ISCA proposé pour les 9 collections.

Colle-ction	MFO			GA			ACO			ISCA		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Re0	71,09	68,5	68,79	71,09	69,37	68,79	66,83	66,23	65,33	68,67	71	89
La1s	73,72	76,02	74,33	74,69	77,16	75,4	71,06	72,9	71,42	76,13	76,88	76,1
La2s	76,77	78,21	76,89	76,9	78,52	77,11	72,62	75,37	72,99	78,23	79,22	78,18
Oh0	76,91	86,33	79,67	76,95	84,98	79,11	71,05	79,23	73,06	78,85	87,19	81,52
Oh5	78,89	86,46	80,51	81,17	85,76	82,58	71,4	78,13	72,81	81,59	86,82	83,42
Oh10	71,14	78,35	72,87	72,23	78,14	73,63	64,68	70,34	65,75	73,05	78,41	74,48
Oh15	73,85	82,25	76,31	74,55	82,2	76,99	67,44	74,35	68,91	76,67	83,92	79,02
Fbis	80,96	79,39	79,58	81,58	79,93	80,24	74,25	73,1	72,73	81,39	80,01	80,2
Trec41	87,17	85,88	86,02	87,06	86,04	85,93	84,56	81,66	82,47	88,23	86,49	86,9
AVG	76,72	80,15	77,22	77,36	80,23	77,75	71,54	74,59	71,72	78,09	81,10	80,98

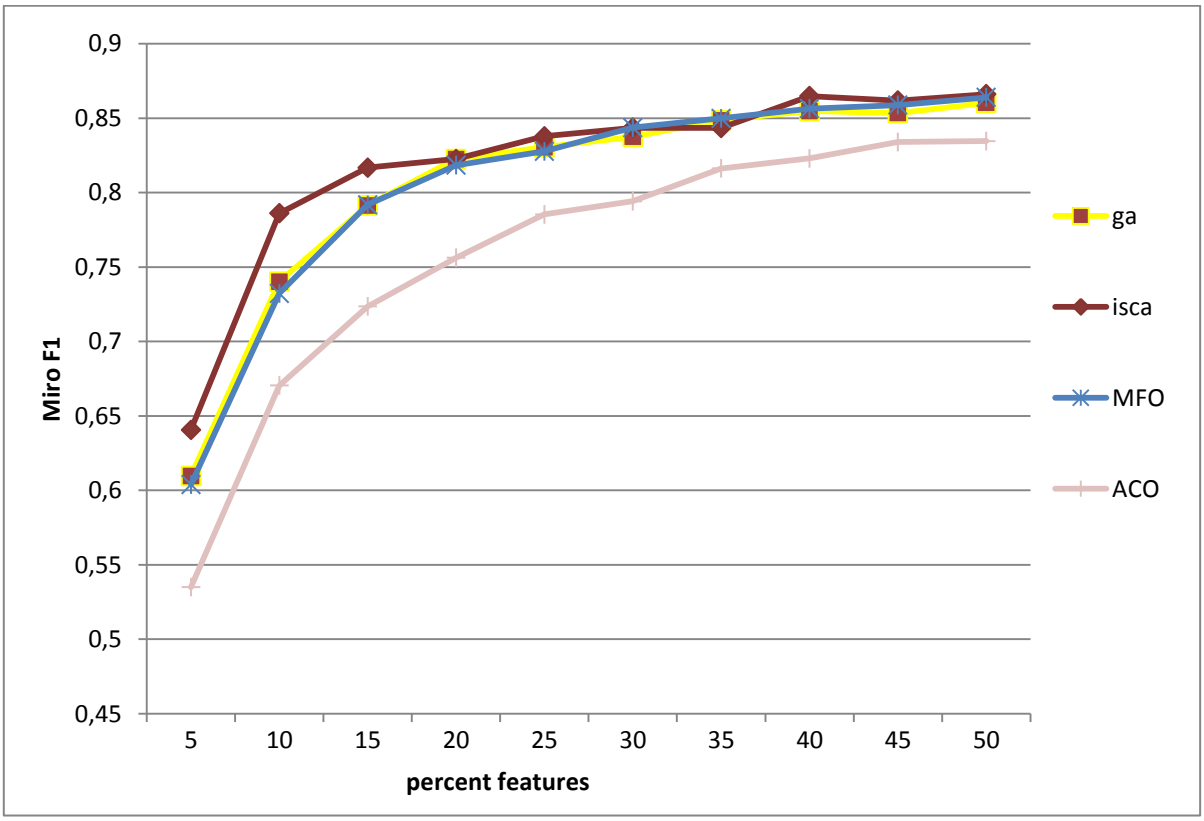
Dans le tableau 5.15, nous présentons les résultats de calcul de moyenne exprimés en micro-F1 et macro-F1 pour les 04 algorithmes utilisés dans ces expériences. Selon ce tableau, il est clair que notre algorithme ISCA proposé surpasse considérablement tous les autres algorithmes en termes de Macro-F1 et Micro-F1 dans la plupart des jeux de données textuelles. Cependant, nous remarquons qu'il existe une compétitivité légère entre les algorithmes GA et l'algorithme MFO.

Tableau 5-15 Micro-F1 et Macro-F1 de SCA, OBL-SCA, Levy_SCA , Weighted_SCA , MFO, ACO, GA et de l'ISCA proposé sur les 9 collections de textes (en pourcentage) .

Collection		Algorithme de recherche			
		MFO	GA	Aco	ISCA
Re0	macrof1	68,79	69,06	65,33	73,97
	microf1	71,93	70,89	69,38	75,46
La1s	macrof1	74,33	75,41	71,42	76,1
	microf1	78,11	78,85	75,63	79,02
La2s	macrof1	76,89	77,1	72,99	78,18
	microf1	80,45	80,23	77,05	81,59
Oh0	macrof1	79,67	79,11	73,06	81,52
	microf1	79,81	79,79	74,68	81,62
Oh5	macrof1	82,61	83,28	75,36	83,89
	microf1	80,51	82,58	72,81	83,42
Oh10	macrof1	72,87	73,63	65,75	74,48
	microf1	76,52	76,56	69,6	77,57
Oh15	macrof1	76,31	76,99	68,91	79,02
	microf1	78,06	78,42	71,66	80,39
Fbis	macrof1	79,58	80,24	72,73	80,2
	microf1	85,93	85,8	79,05	85,83
Trec41	macrof1	86,02	85,93	82,47	86,9
	microf1	90,77	90,58	89,13	91,14

De façon similaire à la comparaison interne, les figures 5.3 et 5.4 présentent graphiquement l'influence du pourcentage de caractéristiques sélectionnées sur la performance des algorithmes de recherche pour la sélection des caractéristiques. En effet, Elle représente les moyennes de F1 en micro- moyenne et macro moyenne sur les 9 collections de textes. L'axe vertical affiche la moyenne de la mesure F1. Cependant, l'axe horizontal présente le pourcentage des nombres de caractéristiques retenues. D'après le graphique, notre algorithme

proposé ISCA a obtenu des performances considérables par rapport aux autres algorithmes : les GAs, MFO et l'algorithme ACO.



5Figure 5-3 Moyennes de Micro F1 de SCA, OBL-SCA, Levy_SCA , Weighted_SCA , MFO, ACO, GA et de l'ISCA proposé pour la sélection d'attributs sur les 9 collections en augmentant le nombre de termes.

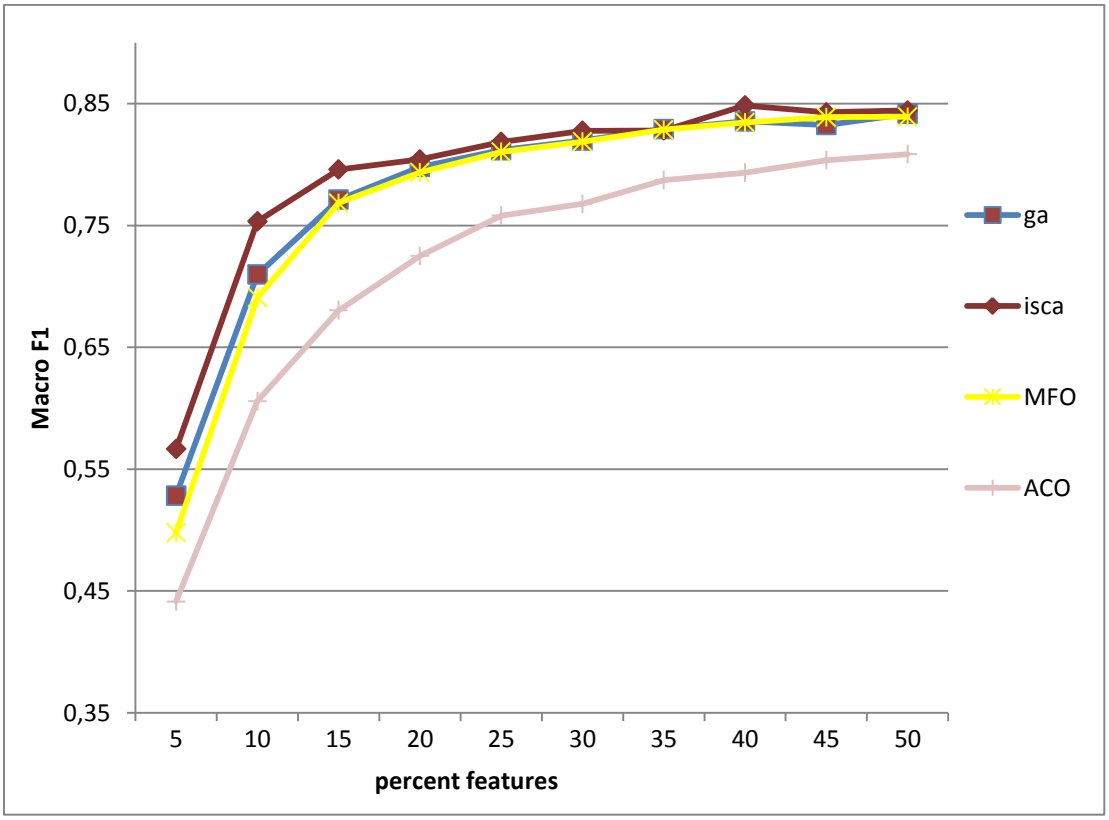


Figure 5-4 Moyennes de MacroF1 de SCA, OBL-SCA, Levy_SCA , Weighted_SCA , MFO, ACO, GA et d'ISCA proposé pour la sélection d'attributs sur les 9 collections en augmentant le nombre de termes.

5.4.3 Comparaison avec les méthodes filtres

En complément, et pour donner plus de crédibilité à notre travail, nous avons comparé les performances de notre algorithme ISCA avec 3 méthodes de sélection d'attributs de type Filtre, à savoir : la méthode IG basée sur le gain d'information (voir section 3.1), la deuxième méthode est la mesure de corrélation (Person), et en dernier, la méthode de signification (Ahmad & Dey, 2005) qui calcule l'importance probabiliste (Probabilistic Significance) d'un attribut. On note ici que la troisième méthode peut servir à la fois pour le classement d'attributs individuellement et/ou pour une sélection d'un sous-ensemble d'attributs. Toutes ces expériences sont réalisées avec l'outil Weka. Le tableau 5.16 présente la haute performance de notre algorithme ISCA en comparaison avec les méthodes filtres citées ci-dessus.

Tableau 5-16 Evaluation de performances de l'algorithme ISCA contre les filtres IG, Corrélation et Importance, sur les 9 collections de texte.

Collection	ISCA Algorithme de recherche			IG Attr . Eval .			corrélation Attr . Eval .			Importance Attr . Eval .		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Re0	82,52	80,28	81,08	70,95	63,10	65,36	50,91	48,98	48,98	48,55	52,54	48,75
la1s	81,40	81,37	81,38	69,04	68,87	68,36	62,07	65,63	62,53	40,56	67,33	40,24
La2s	78,56	77,63	77,81	75,15	74,21	74,27	68,59	67,23	66,63	43,90	67,26	45,34
oh0	82,82	86,52	84,11	80,27	83,25	81,12	64,42	65,15	63,55	63,46	82,02	66,74
Oh5	84,06	85,93	84,71	79,37	80,71	79,75	76,93	80,99	78,09	65,25	90,49	69,83
oh10	74,16	77,69	75,56	70,57	73,50	71,27	69,58	72,15	70,34	55,70	71,60	56,71
Oh15	79,16	84,39	81,12	76,16	79,66	77,28	73,38	76,33	74,46	64,72	80,93	66,14
Fbis	85,86	82,80	83,66	74,40	65,25	65,04	12,86	9,92	9,00	8,21	7,92	7,39
Trec41	92,34	89,39	90,56	74,40	65,25	65,04	64,42	57,16	53,73	59,33	79,84	63,47
Moyen	82,32	82,89	82,22	74,48	72,64	71,94	60,35	60,39	58,59	49,96	66,66	51,62

5.4.4 Le test statistique de *Wilcoxon*

Pour valider statistiquement notre étude et pour lui donner plus de valeur, nous lui avons appliqué le test statistique de *Wilcoxon* (Neuhäuser, 2011). Le test de Wilcoxon-Mann-Whitney (ranksum of Wilcoxon) est un test statistique non paramétrique qui permet de tester l'hypothèse selon laquelle les médianes de chacun de deux groupes de données sont proches. Comme dans tous les tests statistiques, il permet d'accepter et également de rejeter l'hypothèse NULL. Cette dernière considère que la médiane de deux vecteurs des données réelles X et Y est équitable. On outre, ce test supporte même des vecteurs avec des longueurs différentes.

Tableau 5-17 **les valeur de p (** *p* –Values) du Wilcoxon test (ranksum) sur toutes les analyses (les *p*> = 0,05 sont soulignés).

Collection	Algorithme ISCA Vs						
	GA	levySca	OblSca	Sca	WgSca	MFO	ACO
Re0	0,0340	0,0901	0,0000	0,0001	0,0000	0,0266	0,0000
la1s	0,7369	0,5673	0,2145	0,3556	0,0810	0,9707	0,1227
La2s	0,6746	0,5392	0,0876	0,1112	0,0956	0,5426	0,0447
Oh0	0,8719	0.1400	0,0001	0,0002	0,0000	0,3864	0,0000
oh5	0,8786	0,9698	0,0000	0,0000	0,0000	0,3381	0,0000
Oh10	0,8556	0,5609	0,0046	0,0704	0,0012	0,6113	0,0016
oh15	0,2678	0,3481	0,0000	0,0005	0,0000	0,1862	0,0000
FBIS	0,9270	0,4643	0,0050	0,0743	0,0023	0,7085	0,0018
Trec41	0,9348	0,8603	0,0904	0,2008	0,0076	0,6761	0,1234

Le tableau en-dessous montre les valeurs de P du test Wilcoxon. Le test statistique non paramétrique de Wilcoxon (Neuhäuser, 2011) a été appliqué dans cette expérimentation. Selon les valeurs de p, la supériorité de notre algorithme ISCA est statistiquement significative par rapport à celles des algorithmes : Obl –SCA, Weighted-SCA et ACO dans la plupart des collections utilisées. Cependant, cette supériorité est statistiquement faible pour les autres algorithmes tels que GA, LevySca , SCA et MFO .

5.5 **Conclusion**

Nous avons introduit dans ce chapitre une stratégie à deux étapes pour la section des caractéristiques. Notre algorithme de recherche est implémenté dans le processus de sélection des caractéristiques en combinaison avec le filtre du gain d'information. À ce titre, une série d'expériences a été réalisée sur neuf collections de textes composés à partir des bases de données standards telles que Reuters-21578, TREC et OHUSMED. Pour valider notre algorithme, nous avons établi une comparaison avec 07 algorithmes de recherche.

Les résultats expérimentaux montrent les bonnes performances de l'algorithme ISCA proposé pour la sélection de termes. On outre, les méthodes filtres sont aussi comparées avec notre

algorithme ISCA. Finalement, le test statistique de Wilxocon a été appliqué pour montrer la supériorité statistique de notre approche.

Conclusion Générale

6 Conclusion Générale

Le modèle de sac de mots est couramment utilisé dans la catégorisation de textes. Dans ce modèle, les algorithmes d'apprentissage automatique souffrent du problème de la haute dimensionnalité. Le vecteur de caractéristiques (ou d'attributs) produit par la phase prétraitement comporte souvent des centaines de milliers d'attributs. Pour surmonter cette contrainte, on trouve parmi les solutions existantes la sélection des attributs. Plusieurs approches existent dans la littérature traitant ce sujet. Dans cette thèse, nous avons choisi l'approche Enveloppe (Wrapper) qui est l'une des méthodes prometteuse dans la sélection d'attributs. Cependant, le nombre important d'attributs rend la tâche de cette méthode trop lente, éventuellement non pratique ou parfois impossible. Afin de l'adapter à ce problème, nous avons combiné la méthode Wrapper avec une méthode de sélection basée Filtre appliquée en amont. Cette dernière s'appuie sur la mesure statistique gain d'information (IG). Cela permet de de réduire rapidement l'espace de caractéristiques et par conséquent offrir plus d'avantages aux méthodes enveloppes (Wrapper) en termes de coût et notamment de temps d'exécution. Quant à la méthode Wrapper, nous avons proposé un nouvel algorithme de recherche appelé 'Improved Sin Cosine Algorithme' ISCA.

Le nouvel algorithme de recherche ISCA est conçu dans le but, non seulement de réduire la dimensionnalité du vecteur de caractéristiques mais aussi d'améliorer les performances de la phase de classification. Notre algorithme de recherche ISCA est une version améliorée de l'algorithme original SCA. Il se base sur une nouvelle stratégie de déplacement des individus dans l'espace de recherche. Cette amélioration s'exprime par l'insertion d'une nouvelle formule au modèle initial. Plus particulièrement, la mise à jour des positions des solutions est effectuée en tenant compte de deux positions de solutions dans leurs générations : la meilleure solution plus une deuxième position générée aléatoirement. Grâce à cette stratégie, la méthode proposée assure une large exploration du processus de recherche et en même temps converge vers une solution optimale.

En ce qui concerne l'étude expérimentale, nous avons réalisé une série d'expériences sur neuf collections de textes composés à partir de bases de données standards, à savoir : Reuters-21578, TREC et OHUSMED. Les résultats expérimentaux sont très satisfaisants et montrent de bonnes performances et une avance remarquable de notre algorithme proposé ISCA pour FS appliqué au problème de TC. En effet, les résultats ont montré que notre

algorithme ISCA a largement surpassé l'algorithme original SCA ainsi qu'un certain nombre de ses successeurs à savoir : OBL-SCA, Weighted_SCA et Levy_SCA. L'algorithme de recherche MFO est également comparé à ISCA ,ce dérnier le surpasse dans la plupart des 9 collections textuelle utilisées dans cette étude. De plus, nous avons comparé ISCA avec deux autres algorithmes de recherche reconnus et qui ont déjà été utilisés dans ce domaine : les algorithmes génétiques (GA) et les colonies de fourmies (ACO). Là aussi, l'algorithme ISCA atteint vraiment de meilleures performances que GA et ACO et les surpasse largement.
Il est à noter que l'avantage de l'algorithme ISCA proposé ne réside pas seulement dans le fait qu'il a montré de bonnes performances, mais aussi dans sa simplicité de conception puisqu'il repose sur une formule mathématique très simple permettant de mettre à jour les positions des solutions. Le réglage de quelques paramètres rend ISCA assez flexible et facile à adapter à un large éventail de problèmes de recherche, contrairement à d'autres algorithmes, tels qu'ACO et GA, où il est difficile de trouver de nouveaux paramètres appropriés lorsque l'on passe d'un problème de recherche donné à un autre.

En vue de la bonne performance d'ISCA, nous pensons que cet algorithme peut constituer une alternative efficace pour résoudre les problèmes de sélection d'attributs. À l'avenir, nous envisageons d'appliquer l'algorithme ISCA dans d'autres problèmes de sélection d'attributs, et même pour des problèmes complexes dans d'autres domaines. En plus de la résolution du problème de sélection d'attributs, nous souhaitons améliorer encore l'algorithme ISCA et le proposer pour la résolution de problèmes complexes d'optimisation. Enfin, notre algorithme ISCA peut être combiné avec d'autres algorithmes de recherche afin d'améliorer ses performances.

Bibliographie :

Ahmad, A., & Dey, L. (2005). A feature selection technique for classificatory analysis. *Pattern Recognition Letters*, *26*(1), 43–56. https://doi.org/10.1016/j.patrec.2004.08.015

Bakshi, R. K., Kaur, N., Kaur, R., & Kaur, G. (2016). Opinion mining and sentiment analysis. In *Proceedings of the 10th INDIACom; 2016 3rd International Conference on Computing for Sustainable Global Development, INDIACom 2016* (pp. 452–455). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1561/1500000011

Bogunović, A. J. and K. B. and N. (2015). A review of feature selection methods with applications. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on* (pp. 1200–1205).

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1007/bf00058655

Caropreso, M. F., Fernandacnandad, M., Matwin, S., & Sebastiani, F. (2001). A Learner-Independent Evaluation of the Usefulness of Statistical Phrases for Automated Text Categorization.

Carvalho, V. R., & Cohen, W. W. (2005). On the collective classification of email "speech acts." In *SIGIR 2005 - Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 345–352). https://doi.org/10.1145/1076034.1076094

Chakrabarti, S., Dom, B., Agrawal, R., & Raghavan, P. (1997). Using Taxonomy, Discriminants, and Signatures for Navigating in Text Databases. In *In Proceedings of the 23rd VLDB Conference* (pp. 446–455).

Chantar, H. K., & Corne, D. W. (2011). Feature subset selection for Arabic document categorization using BPSO-KNN. In *Proceedings of the 2011 3rd World Congress on Nature and Biologically Inspired Computing, NaBIC 2011* (pp. 546–551). https://doi.org/10.1109/NaBIC.2011.6089647

Cohen, W. W. (1996). Learning Rules that Classify E-Mail.

Cohen, W. W., & Singer, Y. (1999). Context-sensitive Learning Methods for Text Categorization. *ACM Trans. Inf. Syst.*, *17*(2), 141–173. https://doi.org/10.1145/306686.306688

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1023/A:1022627411411

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization (pp. 148–155). Association for Computing Machinery (ACM). https://doi.org/10.1145/288627.288651

Forman, G. (2006). 19MclassTextWc dataset. Retrieved May 25, 2019, from https://sourceforge.net/projects/weka/files/datasets/text-datasets/19MclassTextWc.zip/download

Gilli, Y., & Conde, C. (1988). *Texte et fréquence*. Les Belles Lettres. Retrieved from https://books.google.dz/books?id=2kr1uAEACAAJ

Gomez, J. C., Boiy, E., & Moens, M. F. (2012). Highly discriminative statistical features for email classification. *Knowledge and Information Systems*, *31*(1), 23–53. https://doi.org/10.1007/s10115-011-0403-7

Grandvalet, Y. (2004). Bagging equalizes influence. *Machine Learning*, *55*(3), 251–270. https://doi.org/10.1023/B:MACH.0000027783.34431.42

Hedar, A. R., Wang, J., & Fukushima, M. (2008). Tabu search for attribute reduction in rough set theory. *Soft Computing*, *12*(9), 909–918. https://doi.org/10.1007/s00500-007-0260-1

Hersh, W., Buckley, C., Leone, T. J., & Hickam, D. (1994). OHSUMED: An Interactive Retrieval Evaluation and New Large Test Collection for Research. In B. W. Croft & C. J. van Rijsbergen (Eds.), *SIGIR '94* (pp. 192–201). London: Springer London.

Hull, D. (1994). Improving Text Retrieval for the Routing Problem Using Latent Semantic Indexing. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 282–291). New York, NY, USA: Springer-Verlag New York, Inc. Retrieved from http://dl.acm.org/citation.cfm?id=188490.188585

Jalam, R. (2003). *Apprentissage automatique et catégorisation de textes multilingues*. Retrieved from http://www.theses.fr/2003LYO20056/document

Karabulut, M. (2013). Fuzzy unordered rule induction algorithm in text categorization on top of geometric particle swarm optimization term selection. *Knowledge-Based Systems*, *54*, 288–297. https://doi.org/10.1016/j.knosys.2013.09.020

Kim, Y. H., Hahn, S. Y., & Zhang, B. T. (2000). Text filtering by boosting Naive Bayes classifiers. *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, 168–175. https://doi.org/10.1145/345508.345572

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, *34*(5–6), 975–986. https://doi.org/10.1007/BF01009452

Kogan, J. (2003). Feature Selection and Document Clustering. *Survey of Text Mining*, 73–100. https://doi.org/10.1007/978-1-4757-4305-0_4

L'Huillier, G., Hevia, A., Weber, R., & Ríos, S. (2010). Latent semantic analysis and keyword extraction for phishing classification. In *ISI 2010 - 2010 IEEE International Conference on Intelligence and Security Informatics: Public Safety and Security* (pp. 129–131). https://doi.org/10.1109/ISI.2010.5484762

Lang, K. (1995a). NewsWeeder: Learning to Filter Netnews. In *Machine Learning Proceedings 1995* (pp. 331–339). Elsevier. https://doi.org/10.1016/b978-1-55860-377-6.50048-7

Lang, K. (1995b). NewsWeeder: Learning to Filter Netnews (pp. 331–339). https://doi.org/10.1016/B978-1-55860-377-6.50048-7

Lewis, D. D. (2004). Reuters-21578. Retrieved November 6, 2013, from http://www.daviddlewis.com/%0Aresources/testcollections/reuters21578/.

Lewis, David Dolan. (1992). *Representation and Learning in Information Retrieval*. University of Massachusetts, Amherst, MA, USA.

Li, N., Li, G., & Deng, Z. (2017). An improved sine cosine algorithm based on levy flight (p. 104204R). https://doi.org/10.1117/12.2282076

Li, Q., Wen, Z., & He, B. (2019). Practical Federated Gradient Boosting Decision Trees. Retrieved from http://arxiv.org/abs/1911.04206

Li, Q., Wu, Z., Wen, Z., & He, B. (2019). Privacy-Preserving Gradient Boosting Decision Trees. Retrieved from http://arxiv.org/abs/1911.04209

Lin, K.-C., Hsu, S.-H., & Hung, J. C. (2012). Adaptive SVM-Based Classification Systems Based on the Improved Endocrine-Based PSO Algorithm. In R. Huang, A. A. Ghorbani, G. Pasi, T. Yamaguchi, N. Y. Yen, & B. Jin (Eds.), *Active Media Technology: 8th International Conference, AMT 2012, Macau, China, December 4-7, 2012. Proceedings* (pp. 543–552). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35236-2_55

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, *17*(4), 491–502. https://doi.org/10.1109/TKDE.2005.66

McCallum, A., & Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. *AAAI/ICML-98 Workshop on Learning for Text Categorization*, 41–48. https://doi.org/10.1.1.46.1529

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. https://doi.org/10.1063/1.1699114

Miller, E., Shen, D., Liu, J., & Nicholas, C. (2000). Performance and Scalability of a Large-Scale N-gram Based Information Retrieval System. *Journal of Digital Information; Vol 1, No 5 (2000)*, *1*.

Mirjalili, S. (2015). Moth-Flame Optimization Algorithm: A Novel Nature-inspired Heuristic Paradigm. *Knowledge-Based Systems*, *89*. https://doi.org/10.1016/j.knosys.2015.07.006

Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems*, *96*, 120–133. https://doi.org/10.1016/j.knosys.2015.12.022

Mladeni, D. (2006). Feature Selection for Dimensionality Reduction. *Subspace, Latent Structure and Feature Selection*, *3940*, 84–102. https://doi.org/10.1007/11752790_5

Neuhäuser, M. (2011). Wilcoxon--Mann--Whitney Test. In M. Lovric (Ed.), *International Encyclopedia of Statistical Science* (pp. 1656–1658). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04898-2_615

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137. Retrieved from http://dblp.uni-trier.de/db/journals/program/program14.html#Porter80

Réhel, S. (2005). *Catégorisation automatique de textes et cooccurrence de mots provenant de documents non étiquetés*. *Faculty of Science and Engineering, University LAVAL, QUEBEC*. LAVAL QUÉBEC.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, CSCW 1994* (pp. 175–186).

Association for Computing Machinery, Inc. https://doi.org/10.1145/192844.192905

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Rossi, R., Marcacini, R., & Rezende, S. (2013). Benchmarking Text Collections for Classification and Clustering Tasks. *Technical Report 395, Institute of Mathematics and Computer Sciences - University of Sao Paulo*.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Salton, G., & McGill, M. J. (1986). *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc.

Schmid, H. (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. *Proceedings of International Conference on New Methods in Language Processing*. Retrieved from http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

Scott, S., & Matwin, S. (1999). Feature Engineering for Text Classification. In *Proceedings of ICML-99, 16th International Conference on Machine Learning* (pp. 379–388). Morgan Kaufmann Publishers.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1–47. https://doi.org/10.1145/505282.505283

Shannon, C. E. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal*, *27*(3), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

TREC. (2013). Text REtrieval Conference data. Retrieved November 6, 2013, from http://trec.nist.gov/%0Adata.html.

Uğuz, H. (2011). A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm. *Knowledge-Based Systems*, *24*(7), 1024–1032. https://doi.org/10.1016/j.knosys.2011.04.014

Verbeek, J. (2000). *Supervised feature extraction for text categorization*. Retrieved from https://hal.inria.fr/inria-00321520

Xia, Y., & Wong, K. F. (2006). Binarization approaches to email categorization. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 4285 LNAI, pp. 474–481). https://doi.org/10.1007/11940098_50

Yang, Y, & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Machine Learning-International Workshop Then Conference-* (pp. 412–420). https://doi.org/10.1093/bioinformatics/bth267

Yang, Yiming, & Pedersen, J. O. (1997). A Comparative Study on Feature Selection in Text Categorization. *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, 412–420. https://doi.org/10.1093/bioinformatics/bth267

Zhang, H., & Sun, G. (2002). Feature selection using tabu search method. *Pattern*

Recognition, *35*(3), 701–711. https://doi.org/10.1016/S0031-3203(01)00046-2

Zhang, L., & Liu, B. (2017). Sentiment Analysis and Opinion Mining. In *Encyclopedia of Machine Learning and Data Mining* (pp. 1152–1161). Springer US. https://doi.org/10.1007/978-1-4899-7687-1_907

Zhu, Y., Zhang, B., Wang, Q., Li, W., & Cai, X. (2018). The principle of least effort and Zipf distribution. *Journal of Physics: Conference Series*, *1113*, 12007. https://doi.org/10.1088/1742-6596/1113/1/012007