République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université de Mohamed El Bachir El Ibrahimi de Borj Bou Arréridj
Faculté des Mathématiques et d'Informatique
Département d'informatique

MEMOIRE

Présenté en vue de l'obtention du diplôme

Master en Informatique

Spécialité : Technologies de l'Information et de la Communication (TIC)

THEME

Etude Comparative Des Racinisateurs Arabes.

Présenté par :

MAOUCHE Farah Hiba

KHALFA Lynda

Soutenu publiquement le : 19/06/2024

Devant le jury composé de :

Président : CHARIKHI Mourad

Examinateur : SAIFI Abdelhamid

Encadreur : BELAZZOUG Mouhoub

2023/2024

REMERCIEMENTS

Avant toute chose, nous souhaitons exprimer notre gratitude envers Allah le Tout-Puissant et Miséricordieux, dont la force et la patience nous ont permis d'accomplir ce modeste travail.

Nous souhaitons exprimer nos plus sincères remerciements à notre encadreur Dr BELLAZOUG Mouhoub pour son soutien inestimable, ses conseils éclairés et son dévouement tout au long de l'élaboration de ce travail de recherche. Votre expertise et votre encouragement ont été des piliers essentiels de notre succès.

Nous adressons également nos remerciements à tous les membres du jury, pour avoir accepté d'évaluer notre travail avec attention et impartialité. Vos commentaires constructifs et votre expertise ont grandement enrichi notre réflexion.

Nos remerciements vont également à tous les professeurs qui nous ont guidés, enseignés et inspirés tout au long de notre parcours académique. Votre passion pour l'enseignement et votre engagement envers notre réussite ont été une source constante d'inspiration.

Nous exprimons notre profonde gratitude envers nos familles pour leur soutien indéfectible, leurs encouragements constants et leur compréhension tout au long de ce parcours. Leur amour et leur soutien ont été notre plus grande force.

Nous tenons également à remercier nos amis et nos collègues pour leur soutien, leurs encouragements et leur camaraderie. Leurs conseils, leur présence et leurs encouragements ont rendu ce voyage académique plus enrichissant et mémorable.

Enfin, nous remercions tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail. Votre soutien et votre engagement ont été d'une valeur inestimable et nous vous en sommes profondément reconnaissants.

DÉDICACES

Je dédie également ce travail :

À ma mère et mon père, pour leur soutien inébranlable, leurs encouragements constants et leur amour inconditionnel.

À mon grand-père, pour son amour, sa sagesse et ses précieux conseils tout au long de ma vie.

À mes frères, Mohamed, Mahdi et Abdelhak, pour leur soutien fraternel, leurs encouragements et leur aide précieuse.

À mes amies, Ichrak, Nour, Farah, Khouloud, Lyna et Chaima, pour leur compréhension, leur motivation et les moments de détente qui m'ont permis de garder le cap.

À mon binôme, Lynda, pour sa collaboration, son dévouement et son esprit d'équipe. Ensemble, nous avons surmonté les défis et réalisé ce travail avec succès.

À mon encadrant, Belaazoug Mouhoub, pour son orientation, ses conseils précieux et sa patience tout au long de ce parcours.

Et enfin, à toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce travail. Merci pour votre soutien et votre bienveillance.

MAOUCHE Farah Hiba

DÉDICACES

Je dédie ce modeste travail :

A mon père, que son âme repose en paix.

A ma chère mère, ma grand-mère, mes oncles et mes tantes, qui ont toujours été là pour me soutenir et m'encourager dans toutes les étapes de ma vie.

A mes chères cousines, Feriel, Yasmine et Célia, qui ont partagé avec moi tant de moments précieux et qui ont été une source de joie et de réconfort.

A mes amies, Lyna et Chaima, dont le soutien indéfectible et les encouragements ont été un pilier essentiel tout au long de ce parcours.

A mon ami Mohammed, dont la présence et les conseils avisés ont été d'un grand réconfort dans les moments difficiles.

A mon binôme, Farah, avec qui j'ai partagé tant de défis et de réussites, et qui a été un partenaire précieux dans la réalisation de ce travail.

A mon encadreur, Dr. Belazzoug Mouhoub, dont l'expertise, les conseils et le soutien ont été d'une importance capitale pour la réussite de ce projet.

Enfin, à toutes les personnes qui m'ont apporté leur aide, leur soutien et leur encouragement, je vous adresse ma plus profonde gratitude. Ce travail est le fruit de notre collaboration et de notre engagement commun envers l'excellence.

KHALFA Lynda

Résumé

La langue arabe présente une structure morphologique complexe, ce qui constitue un défi unique dans le traitement du langage naturel (NLP). Le système dérivationnel de l'arabe est basé sur des racines, qui sont fréquemment modifiées pour créer de nouveaux mots, en utilisant un ensemble étendu d'affixes morphémiques arabes tels que des préfixes, des suffixes et plus encore. La racinisation est une tâche fondamentale dans le traitement du texte, joue un rôle crucial dans la récupération de l'information (IR) et l'analyse de texte, elle réduit les mots à leur forme de base ou racine, facilitant la normalisation du texte pour un traitement plus facile. Cependant, aucun algorithme de racinisation pour cette langue n'est parfait.

Dans ce travail, nous allons nous concentrer sur la comparaison et l'évaluation des performances de plusieurs raciniseurs arabes, à savoir ISRI, Tashaphyne et Snowball. Nous avons l'intention d'évaluer leurs performances sur deux ensembles de données distincts en utilisant des techniques avancées telles que les réseaux neuronaux et les classificateurs d'apprentissage automatique. De plus, nous visons à déterminer quelle combinaison de raciniseur et de classificateur donne les meilleurs résultats, fournissant des informations inestimables pour les applications de traitement de texte en arabe.

Mot clés : Le traitement du langage naturel, racinisation, Algorithme de classification, Approches, Intelligence artificielle.

Abstract

Arabic language has a complex morphological structure, which presents a unique challenge in natural language processing (NLP). The derivational system of Arabic is based on roots, which are frequently modified to create new words, employing an extensive set of Arabic morphemes affixes such as prefixes, suffixes and more. Stemming is a fundamental task in text processing, plays a crucial role in information retrieval (IR) and text analysis, it reduces words to their basic or root form, facilitating text normalization for easier processing. However, no stemming algorithm for this language is perfect.

In this work, we are going to focus on comparing and evaluating the performance of several Arabic stemmers namely, ISRI, Tashaphyne and Snowball. We intend to assess their performance on two distinct datasets using advanced techniques such as neural networks and machine learning classifiers. Additionally, we aim to determine which combination of stemmer and classifier yields the best results, providing invaluable insights for Arabic text processing applications.

Keywords: Natural language processing, stemming, Classification algorithm, Approaches, Artificial intelligence.

ملخص

اللغة العربية لديها هيكل تصريفي معقد، مما يشكل تحديًا فريدًا في معالجة اللغة الطبيعية. يستند النظام التصريفي للعربية على الجذور، التي يتم تعديلها بشكل متكرر لإنشاء كلمات جديدة، باستخدام مجموعة واسعة من البادئات واللاحقات الصرفية العربية. التجذيع هو مهمة أساسية في معالجة النصوص، ويؤدي دورًا حاسمًا في استرجاع المعلومات وتحليل النصوص، حيث يقلل الكلمات إلى شكلها الأساسي أو الجذري، مما يُيسّر التعامل مع النصوص. ومع ذلك، لا يوجد خوارزمية تجذيع مثالية لهذه اللغة.

في هذا العمل سنركز على مقارنة وتقييم أداء عدة مقلمات عربية ونعتزم على تقييم أدائها على مجموعتي بيانات متميزتين باستخدام تقنيات متقدمة مثل الشبكات العصبية والمصنفات الآلية. بالإضافة إلى ذلك، نهدف إلى تحديد أي توليفة من المجذع والمصنف تعطي أفضل النتائج، مما يوفر رؤى قيمة لتطبيقات معالجة النصوص العربية.

الكلمات الرئيسية: معالجة اللغة الطبيعية، التجذيع، خوارزمية التصنيف، النهج، الذكاء الاصطناعي.

Table des matières

Table des figures

Liste des tableaux

Liste d'abréviations

DL	Deep Learning
ML	Machine Learning
SVM	Support Vector Machine
NB	Naïve bayes
MLP	Multilayer Perceptron
GB	Gradient Boosting
ACC	Accuracy
DT	Arbres de décision
AI	Intelligence Artificielle

Introduction Générale

Introduction générale

- ✓ **Contexte de travail**

La langue arabe, riche de son histoire et de sa culture millénaire, est une des plus anciennes langues écrites du monde. Cette longue histoire a produit une énorme quantité de documents textuels, allant des manuscrits anciens aux publications contemporaines. En outre, avec l'avènement du numérique, la production de contenus en arabe a explosé, englobant des livres, des articles scientifiques, des articles de presse, des blogs, des forums, et des réseaux sociaux. Ainsi, nous sommes confrontés à une vaste toile de données textuelles arabes, chacune renfermant un savoir précieux, des informations historiques et culturelles, ainsi que des connaissances scientifiques et techniques.

Cependant, ce flot incessant d'informations présente des défis importants en termes d'organisation et d'exploitation. La langue arabe, avec sa richesse morphologique, sa variété dialectale et ses structures grammaticales complexes, rend le traitement automatique de textes particulièrement difficile. La diversité des sources et des styles de rédaction ajoute une couche de complexité supplémentaire. Face à cette abondance de données, il devient essentiel de disposer d'outils efficaces pour trier, organiser et accéder rapidement aux informations pertinentes.

C'est dans ce contexte que la classification automatique des textes arabes devient un outil indispensable. Cette technologie permet d'assigner automatiquement des catégories pertinentes aux documents écrits en arabe, facilitant ainsi leur organisation, leur recherche et leur analyse. Grâce à la classification automatique, il est possible de naviguer plus facilement dans cet océan de connaissances, d'extraire rapidement des informations utiles, et de tirer des enseignements significatifs de vastes ensembles de données textuelles. La classification automatique ouvre également des perspectives nouvelles pour l'exploration de la littérature arabe, la préservation de son patrimoine et la diffusion de ses savoirs à un public plus large.

- ✓ **Problématique**

L'immensité et la diversité des données textuelles en arabe posent des défis spécifiques en termes d'organisation et d'exploitation efficaces. La langue arabe présente une série de particularités linguistiques qui compliquent le traitement automatique des textes. Parmi ces

particularités, on trouve une morphologie riche, avec des mots pouvant prendre de nombreuses formes différentes à partir d'une même racine. L'arabe est également caractérisé par un système complexe de dérivation et d'inflexion, où un mot peut varier en fonction du contexte grammatical, du temps, de l'aspect et de la voix. Cette flexibilité rend la tâche de reconnaissance et de classification automatique des mots et des textes particulièrement ardus.

En outre, l'arabe utilise un script cursif, où les lettres changent de forme en fonction de leur position dans le mot, ce qui peut compliquer davantage la segmentation des mots et des phrases. Les variations dialectales ajoutent une autre couche de complexité : l'arabe moderne standard diffère des dialectes régionaux, qui peuvent varier considérablement d'un pays à l'autre, voire d'une région à une autre au sein du même pays. Ces différences dialectales peuvent affecter la lexique, la syntaxe et même la sémantique des textes.

Face à ces défis, il est crucial de développer des méthodes robustes pour assigner automatiquement des catégories pertinentes aux documents écrits en arabe. La question centrale est donc de savoir comment on peut efficacement réaliser cette tâche. Quels algorithmes de classification sont les plus adaptés pour traiter des textes arabes ? Les techniques de traitement préalable, comme le stemming (racinisation) ou la lemmatisation, peuvent-elles améliorer la précision de la classification ? Comment gérer les variations linguistiques et dialectales pour optimiser les performances des systèmes de classification ?

Pour répondre à ces questions, il est nécessaire d'explorer différentes approches et techniques. Par exemple, certains algorithmes de classification, comme les machines à vecteurs de support (SVM), les réseaux de neurones, ou les modèles basés sur l'apprentissage profond, pourraient offrir des performances variées en fonction des caractéristiques spécifiques des données arabes. De plus, l'efficacité de ces algorithmes peut être influencée par la qualité du prétraitement des données. Des techniques comme la tokenisation, le filtrage des mots vides (stop words), et la normalisation des caractères doivent être adaptée aux particularités de la langue arabe.

Ainsi, la problématique centrale de cette étude est de déterminer les meilleures pratiques pour la classification automatique des textes arabes, en prenant en compte les défis linguistiques uniques posés par cette langue. Cela implique d'évaluer l'efficacité de divers algorithmes et techniques de prétraitement pour optimiser les performances de classification dans divers contextes et types de données textuelles en arabe.

- ✓ **Objectif**

L'objectif principal de cette étude est d'explorer en profondeur les principes fondamentaux de la classification des textes en arabe, en examinant comment différents algorithmes et techniques de prétraitement influencent les performances de cette classification. La complexité de la langue arabe, avec ses particularités morphologiques et syntaxiques, exige une approche méthodique pour comprendre et surmonter les défis spécifiques liés au traitement automatique des textes.

Pour atteindre cet objectif, nous allons d'abord étudier les concepts de base de la classification des textes, y compris les différentes étapes du processus, telles que la préparation des données, le choix des caractéristiques (features) et la sélection des algorithmes de classification. Nous chercherons à identifier quelles techniques de prétraitement, comme la tokenisation, le filtrage des mots vides (stop words), et surtout la racinisation (stemming), sont les plus efficaces pour améliorer la qualité de la classification des textes en arabe.

Ensuite, nous évaluerons l'efficacité de divers algorithmes de classification dans des contextes variés. Ces algorithmes incluront des méthodes traditionnelles, comme les machines à vecteurs de support (SVM), et les Naïve Bayes (NB), ainsi que des techniques plus avancées d'apprentissage profond, telles que Multi-layer Perceptron (MLP) et les Gradient Boosting. Nous comparerons leurs performances en termes de précision, de rappel, et de F-mesure sur différents ensembles de données arabes.

Un aspect crucial de notre étude sera l'analyse de l'impact de la variation linguistique sur les résultats de la classification. Nous utiliserons différents stemmers pour la langue arabe, tels que le stemmer de ISRI, Snowball et le Tashaphyne, afin d'observer comment ces techniques affectent la performance des algorithmes de classification. Cette comparaison nous permettra de mieux comprendre quelles méthodes de racinisation sont les plus appropriées pour divers types de textes et contextes linguistiques.

En résumé, cette étude vise à fournir une analyse exhaustive des techniques de classification des textes arabes, en tenant compte des spécificités linguistiques et des défis posés par cette langue. Nous espérons que nos recherches permettront de déterminer les meilleures pratiques pour le traitement et la classification des textes arabes, facilitant ainsi l'accès à l'information et la découverte de nouvelles connaissances dans ce riche corpus textuel.

- ✓ **Plan de mémoire**

Pour mener à bien cette mission, nous structurerons notre recherche en quatre chapitres distincts.

Le premier chapitre, intitulé "Classification de textes automatique", explorera les étapes clés de la classification de textes, y compris le prétraitement, les applications pratiques, les défis rencontrés et les algorithmes utilisés.

Le deuxième chapitre se concentrera sur la racinisation (stemming), une technique visant à réduire les mots à leur racine commune, avec une attention particulière aux techniques spécifiques à la langue arabe.

Le troisième chapitre détaillera la méthodologie employée dans la recherche, couvrant la collecte des données et le déroulement de l'étude.

Enfin, le quatrième chapitre présentera les résultats de la recherche, décrivant les outils utilisés, analysant les résultats obtenus et synthétisant les conclusions tirées de l'étude.

En conclusion, ce mémoire aspire à apporter une contribution notable au domaine de la catégorisation automatique des textes arabes, facilitant l'accès à l'information et la découverte de connaissances dans ce riche univers textuel. Notre travail se distingue par son originalité dans sa combinaison d'une analyse approfondie des techniques de classification et de racinisation pour la langue arabe.

Chapitre 01

Classification automatique de textes

1.1. Introduction

Ce premier chapitre est consacré à une étude détaillée de la classification des textes et couvre de nombreux aspects fondamentaux. Commençons par prétraiter les données texte. Il s'agit d'une étape essentielle pour garantir la qualité et la pertinence des données avant analyse. Nous examinons ensuite diverses applications réelles de la classification de texte et démontrons son utilisation dans divers domaines tels que l'analyse des sentiments, le filtrage du spam et la classification automatique des documents. Nous abordons également les principaux défis dans le domaine, notamment la résolution des problèmes liés aux ensembles de données déséquilibrés, à la complexité du traitement du langage naturel et aux performances des modèles. De plus, nous présentons une étude détaillée des principaux algorithmes de classification, discutons de leur fonctionnement, ainsi que de leurs avantages et limites. L'objectif principal de ce chapitre est de fournir un aperçu complet et nuancé des principes et techniques de classification de textes, en mettant l'accent sur les défis actuels et les opportunités futures dans ce domaine en évolution.

1.2. Classification de textes

La classification de textes est le processus d'attribution d'un texte libre à une catégorie (ou classe) spécifique en fonction de son contenu. Pour ce faire, un ensemble de textes préalablement étiquetés, appelé ensemble d'apprentissage, est préparé. Cet ensemble sert à générer le modèle de prédiction le plus adéquat et performant possible.

Formellement, le problème de classification des textes (TC) consiste à assigner une valeur booléenne (Vrai ou Faux) à chaque paire (di, ci) ∈ D x C, où "Vrai" signifie que le document appartient à la catégorie et "Faux" qu'il n'y appartient pas. Dans cette notation, D = {d1, ..., dn} représente l'ensemble des documents à catégoriser et C = {c1, ..., cm} désigne les catégories de texte prédéfinies. (1)

1.3. Processus de classification de textes

Le processus de catégorisation consiste à construire un modèle de prédiction qui, en prenant un texte en entrée, lui associe une ou plusieurs étiquettes en sortie. Pour déterminer la catégorie ou la classe d'un texte, plusieurs étapes sont généralement suivies. Ces étapes incluent la représentation du texte, le choix de l'algorithme d'apprentissage, et l'évaluation des résultats pour garantir une bonne généralisation du modèle.

L'apprentissage, qui comprend plusieurs étapes et aboutit à un modèle de prédiction, se déroule comme suit (1) :

a. Un ensemble de textes étiquetés est disponible, où chaque texte est associé à une catégorie connue.

b. A partir de ce corpus, les descripteurs (ou mots, ou termes) les plus pertinents pour le problème à résoudre sont extraits.

c. Un tableau des descripteurs × individus est constitué, où chaque texte est représenté par ses descripteurs et son étiquette.

d. Un algorithme d'apprentissage est appliqué à ce tableau pour obtenir un modèle de prédiction.

Ce modèle peut ensuite être utilisé pour classer de nouveaux textes en leur attribuant les étiquettes appropriées.

Le classement d'un nouveau texte dx se déroule en deux étapes :

a. Rechercher et pondérer les occurrences des termes (t1, ..., tk) dans le texte dx à classer.

b. Appliquer le modèle sur ces occurrences pour prédire l'étiquette de ce texte dx.

Le processus de catégorisation, est résumé dans la figure 1 :

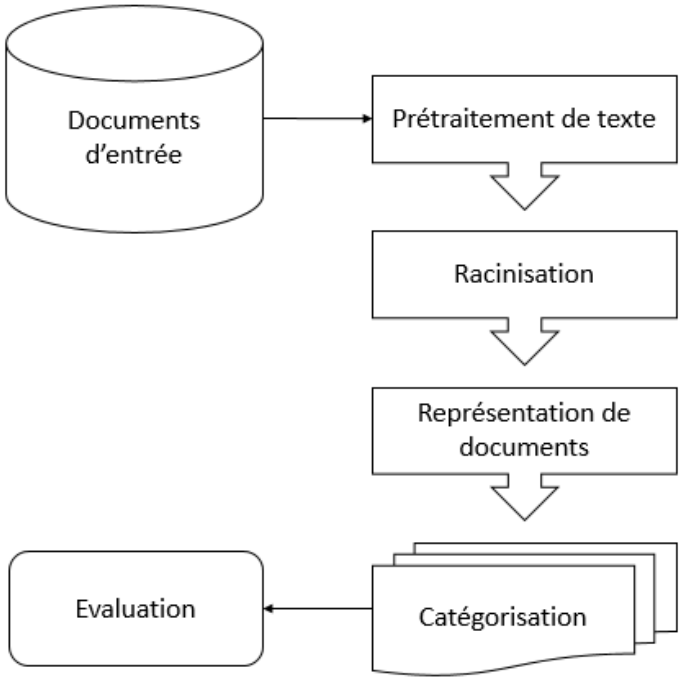

Figure 1 Processus de catégorisation automatique de textes.

Le processus de catégorisation des textes repose sur l'utilisation de techniques d'apprentissage automatique pour attribuer des catégories à des documents textuels. Ce processus commence par le prétraitement des données, qui comprend le nettoyage et la normalisation des textes. Les textes sont ensuite représentés sous forme de vecteurs numériques. Un modèle est alors entraîné sur un ensemble de données annoté, apprenant à associer les caractéristiques des textes à leurs catégories respectives. Après des phases de validation et d'ajustement, le modèle est testé sur un ensemble de données indépendant pour évaluer sa précision. Une fois validé, le modèle est déployé en production pour catégoriser de nouveaux documents, avec une surveillance et une maintenance régulières pour assurer la pérennité de ses performances.

1.4. La représentation des textes

La représentation des textes est une étape cruciale dans le processus de classification, nécessitant l'utilisation d'une technique efficace pour rendre les textes exploitables par la machine. Le modèle le plus couramment utilisé est le modèle vectoriel, où chaque texte est représenté par un vecteur de n termes pondérés. Les différentes techniques de représentation des textes incluent (1) :

- ✓ Sac de mots (Bag-of-Words) : représente les mots par leur fréquence.
- ✓ TF-IDF : attribue un poids aux mots en fonction de leur importance.
- ✓ Word Embeddings : représente les mots par des vecteurs capturant leurs relations sémantiques.
- ✓ Modèles de langage pré-entraînés : fournissent des représentations de haute qualité pour les mots et les phrases.
- ✓ N-grammes : capture les séquences de mots consécutifs.

1.5. Le prétraitement

Avant de procéder à l'encodage vectoriel, il est indispensable de réaliser une étape préliminaire de transformation d'un document depuis l'espace des caractères vers l'espace des mots. Cette opération, connue sous le nom de prétraitement, prépare ainsi les données pour leur représentation vectorielle (1).

1.5.1. La segmentation

La phase de segmentation implique de découper la séquence des caractères afin de regrouper les caractères formant un même mot. Une méthode classique de segmentation consiste à découper les séquences de caractères en fonction de la présence ou de l'absence de caractères de séparation tels que les espaces, les tabulations ou les retours à la ligne.

1.5.2. Le traitement morphologique

Le traitement morphologique implique de traiter individuellement chaque mot afin de regrouper les mots qui sont significativement identiques.

- **Élimination de mots vides :** Certains mots tels que les prépositions, conjonctions ou articles, qui n'apportent aucune information sémantique et ne modifient pas le sens des mots qui les accompagnent, peuvent être retirés des documents. Cette étape de prétraitement permet de réduire la taille du lexique.
- **Le stemming :** Cette méthode vise à regrouper sous un même identifiant des mots partageant une racine commune.
- **Lemmatisation :** La lemmatisation implique de regrouper des mots ayant la même signification même si leurs racines diffèrent (par exemple, "maison" et "baraque"). Cette tâche est plus complexe que le stemming et repose souvent sur l'utilisation de vastes bases de connaissances. Par exemple, le lemme d'un mot au pluriel sera sa forme au singulier, et celui d'un verbe conjugué sera sa forme infinitive.

1.5.3. Le traitement syntaxique

- **Sélection des attributs** : Les critères les plus utilisés sont :

- **La fréquence documentaire :** Cette méthode consiste à éliminer les mots dont la fréquence d'apparition dans les documents est inférieure à un seuil donné. L'objectif est de supprimer les mots peu informatifs ou susceptibles d'être des erreurs typographiques. Bien que simple et rapide, cette technique n'est généralement pas utilisée de manière agressive, car les termes ayant une fréquence faible à moyenne sont considérés comme relativement informatifs et devraient être conservés.

- **Le gain d'information :** Cette méthode évalue le pouvoir discriminant d'un mot en mesurant le nombre de bits d'information obtenus pour prédire la catégorie en fonction de la présence ou de l'absence du mot. Elle est souvent utilisée dans la construction d'arbres de décision pour choisir l'attribut qui divise le mieux l'ensemble des instances en deux groupes homogènes.

- **La force du terme :** Cette méthode est différente des précédentes. Elle vise à estimer l'importance d'un terme en fonction de sa tendance à apparaître dans des documents similaires. Elle commence par former des paires de documents dont la similarité cosinus est supérieure à un seuil donné. Ensuite, la force d'un terme est calculée en utilisant la probabilité conditionnelle qu'il apparaisse dans le deuxième document d'une paire, sachant qu'il apparaît dans le premier.

- **Tableau « individus – variables » :** il s'agit de transformer l'ensemble des textes en un tableau croisé où :
  - Chaque individu représente un texte étiqueté pendant la phase d'apprentissage et à classer lors de la phase de prédiction.
  - Les variables sont les descripteurs (ou termes) extraits des données textuelles.
  - Les éléments du tableau représentent le poids de chaque terme dans chaque texte.

1.6. Application de la classification de textes

La classification de textes trouve une multitude d'applications dans divers domaines. Voici quelques exemples de secteurs où elle est couramment utilisée (2) :

- **Filtrage et organisation des nouvelles :**

La majorité des services d'information sont désormais de nature électronique, générant chaque jour un grand nombre d'articles de presse. Organiser ces articles manuellement devient ainsi une tâche ardue. La classification automatique s'avère très utile pour la catégorisation des divers portails web, également appelée filtrage de textes, comme mentionné dans les travaux de Lang (1995a) et Resnick et al. (1994).

- **Détection de spam :**

La détection de spam utilise la classification des textes pour identifier automatiquement les courriels indésirables. Les courriels sont prétraités, représentés numériquement, et un modèle

de classification est entraîné sur des données contenant à la fois des courriels spam et légitimes. Une fois le modèle validé, il est utilisé pour analyser en temps réel les courriels entrants et les classer comme spam ou non spam.

- **Analyse de sentiments :**

Application de la classification des textes, identifie l'émotion ou l'attitude exprimée dans un texte, comme positive, négative ou neutre. Les textes sont prétraités et convertis en représentations numériques. En utilisant un modèle de classification entraîné sur des données annotées, le sentiment est prédit dans de nouveaux textes. Cette analyse est cruciale pour comprendre les opinions des clients, la perception des produits et services, et pour orienter les stratégies commerciales.

- **Détection de fake news :**

Dans la détection de fake news, la classification des textes est utilisée pour distinguer les articles de presse authentiques des informations trompeuses. Les textes sont prétraités, convertis en représentations numériques, puis un modèle est entraîné sur des données annotées pour prédire l'authenticité des articles. Cette analyse vise à promouvoir la vérité et la fiabilité de l'information en ligne, contribuant ainsi à une société mieux informée et plus critique.

- **Sécurité et surveillance :**

La sécurité et la surveillance utilisent la classification des textes pour analyser les communications électroniques et identifier les activités suspectes ou criminelles. Les textes sont prétraités, représentés numériquement, puis un modèle est entraîné sur des données annotées pour détecter les menaces potentielles. Cette analyse permet de surveiller en temps réel les conversations et les communications, aidant ainsi les agences gouvernementales et les entreprises de sécurité à prévenir les incidents et à protéger les individus et les organisations contre les dangers en ligne.

1.7. Obstacles dans la classification de textes

La classification de textes, un domaine clé de l'analyse de données textuelles, consiste à attribuer automatiquement des étiquettes ou des catégories à des documents textuels en fonction de leur contenu. Cette tâche revêt une importance croissante dans de nombreux domaines, allant de la recherche d'information à la surveillance en ligne, en passant par la recommandation de contenu personnalisé. Cependant, malgré les avancées significatives réalisées dans ce domaine, la classification de textes est confrontée à plusieurs défis inhérents qui peuvent affecter sa précision et son efficacité.

Dans la continuation, nous abordons certains défis inhérents à la classification automatique de textes (2).

- **La haute dimensionnalité :**

Le modèle de sac de mots génère fréquemment des quantités considérables de termes, souvent des centaines de milliers. Cette abondance peut affecter la qualité de la classification,

engendrant non seulement des coûts en temps liés aux algorithmes de classification, mais également ce que l'on appelle communément un sur-apprentissage. De plus, notre vecteur de mots, caractérisé par une grande majorité de variables vides, présente une structure creuse, ajoutant ainsi un problème supplémentaire à celui de la dimensionnalité et du sur-apprentissage.

- **Complexité de l'algorithme :**

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

- **Variabilité linguistique :**

La diversité des langues, des dialectes et des styles d'écriture peut rendre la classification des textes plus complexe. Les modèles doivent être capables de généraliser à différentes expressions linguistiques pour assurer leur efficacité dans des contextes variés.

- **Classes déséquilibrées :**

Certains jeux de données peuvent présenter un déséquilibre entre les classes, où une classe est significativement plus représentée que d'autres. Cela peut entraîner un biais dans les prédictions du modèle en faveur de la classe majoritaire, réduisant ainsi sa capacité à reconnaître les classes minoritaires.

- **Ambiguïté (polysémie) :**

Dans le langage naturel, un mot unique peut revêtir plusieurs significations différentes, avec plusieurs définitions qui lui sont associées. Par exemple, le mot "jaguar" peut faire référence à l'animal ou à la voiture. Si ces significations ne sont pas distinguées et sont représentées par la même variable (terme), cela risque inévitablement de compromettre la qualité de la classification par la suite.

- **Synonymie :**

Les synonymes sont des mots ou des expressions ayant le même sens. Ne pas tenir compte de ces synonymes lors de la phase de représentation des documents conduira à un élargissement supplémentaire de notre vecteur de mots, entraînant ainsi une baisse de la précision des performances.

- **Présence de bruit dans les données :**

Les textes peuvent contenir des éléments indésirables tels que des fautes d'orthographe, des abréviations, des caractères spéciaux ou des données incomplètes. Ce bruit peut compromettre la qualité des prédictions et nécessite un nettoyage préalable des données.

1.8. Les algorithmes de classification

L'utilisation d'algorithmes de classification est indispensable dans de nombreux domaines de l'informatique et de l'analyse de données. Ces algorithmes permettent de catégoriser

automatiquement des données en fonction de leurs caractéristiques. Dans le domaine de l'analyse de textes, plusieurs de ces algorithmes sont appliqués pour classer efficacement des documents en différentes catégories ou étiquettes.

Dans les sections suivantes, nous présenterons quelques-uns de ces algorithmes de classification largement utilisés dans le domaine de la classification de textes, reconnus pour leur efficacité dans divers domaines d'application.

1.8.1. Algorithme de Rocchio

L'algorithme de Rocchio est une méthode de classification supervisée utilisée principalement dans le domaine de la recherche d'information et de l'apprentissage automatique. Il est principalement utilisé pour classer des documents textuels dans des catégories prédéfinies.

L'idée principale derrière l'algorithme de Rocchio est de représenter chaque classe par un vecteur moyen (appelé centroïde) dans un espace de caractéristiques. Lorsqu'un nouveau document est introduit, il est comparé aux centroïdes des différentes classes et classé dans la classe dont le centroïde est le plus proche.

La formule de Rocchio constitue une extension du modèle vectoriel qui automatise la transformation d'une requête initiale (représentée par un vecteur Q0) en une nouvelle requête (représentée par un vecteur Q1) plus performante. Grâce à ce modèle, un ensemble de documents répondant à la requête initiale est proposé à l'utilisateur qui les étiquette (feedback de pertinence).

La nouvelle requête Q1 est construite selon la formule de Rocchio [Rocchio, 1971], qui consiste à ajouter à la requête initiale les termes des documents pertinents tout en retirant ceux des documents non pertinents (2).

$$\mathbf{Ci}=\boldsymbol{\alpha}\cdot\mathbf{ci}+\boldsymbol{Ni\beta}\sum\mathbf{xj}\in\boldsymbol{Di}\mathbf{xj}-\boldsymbol{N}-\boldsymbol{Ni\gamma}\sum\mathbf{xj}\in\boldsymbol{D}\neg\boldsymbol{i}\mathbf{xj}$$

Où :

$\alpha\alpha$ est le facteur d'atténuation, contrôlant l'influence des anciens centroïdes sur le nouveau.

$\beta\beta$ est le facteur d'importance des documents de la classe, permettant de mettre à jour le centroïde en fonction des exemples positifs.

$\gamma\gamma$ est le facteur d'importance des documents hors classe, permettant de mettre à jour le centroïde en fonction des exemples négatifs.

NiNi est le nombre de documents dans la classe ii.

NN est le nombre total de documents.

xjxj est le vecteur représentant le j-ème document.

DiDi est l'ensemble des documents de la classe ii.

$D\neg iD\neg i$ est l'ensemble des documents hors de la classe ii.

1.8.2. Arbres de décision

Un arbre de décision est un modèle simple où, face à plusieurs caractéristiques, la décision commence par l'une d'entre elles ; si cette première caractéristique n'est pas suffisante, une autre est utilisée, et ainsi de suite. Ce modèle est largement reconnu et utilisé dans de nombreuses entreprises pour faciliter la prise de décision et l'analyse des risques. Initialement populaire dans les années 1960-1980 pour la construction de systèmes experts, son utilisation a décliné lorsque les règles devaient être introduites manuellement. Toutefois, l'émergence de méthodes mathématiques pour construire les arbres de décision a ravivé l'intérêt pour ce modèle dans le domaine de l'apprentissage automatique (3).

Le processus général de création d'un arbre de décision comprend les étapes suivantes :

- Identifier la meilleure caractéristique dans l'ensemble de données d'entraînement.
- Diviser les données d'entraînement en sous-ensembles contenant les différentes valeurs de la meilleure caractéristique.
- Générer de nouveaux arbres de décision de manière récursive en utilisant les sous-ensembles de données créés.
- Arrêter lorsque les données ne peuvent plus être classifiées.

Plusieurs algorithmes automatiques sont disponibles pour construire des arbres de décision. Parmi les plus reconnus figurent les algorithmes ID3, C4.5, J48 et CART, parmi d'autres. La Figure 2 ci-dessous illustre un arbre de décision généré par l'algorithme J48 à partir de la base de données weather.arff.

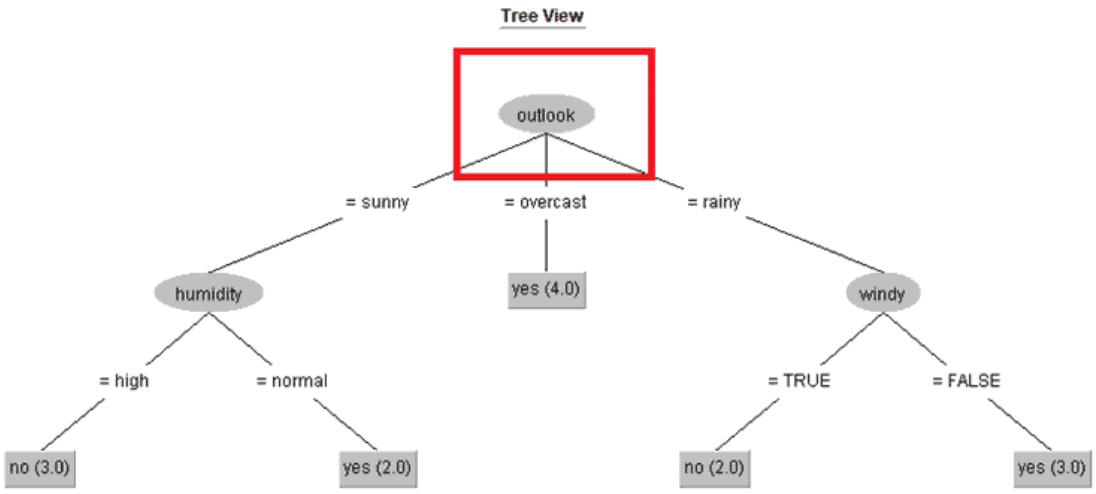

Figure 2 Arbre de décision appliqué sur la base weather.arff avec l'outil Weka.

1.8.3. L'algorithme K-plus proches voisins

L'algorithme des k plus proches voisins, souvent abrégé en KNN ou k-NN, est un outil d'apprentissage supervisé non paramétrique qui se base sur la proximité entre les données pour effectuer des classifications ou des prédictions sur un point de données individuel. Bien qu'il puisse être utilisé pour des problèmes de régression ou de classification, il est le plus souvent employé comme algorithme de classification, partant du principe que des points similaires sont susceptibles d'être regroupés ensemble (4).

- Étape 1 : Choisissez le nombre K de voisins.
- Étape 2 : effectuer le calcul des distances.

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

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

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

Du point non classifié aux autres points.

- Étape 3 : Sélectionnez les K voisins les plus proches en fonction de la distance calculée.
- Étape 4 : Parmi ces K voisins, déterminez le nombre de points appartenant à chaque catégorie.
- Étape 5 : Assignez le nouveau point à la catégorie la plus représentée parmi ces K voisins. Étape 6 : Le modèle est prêt.

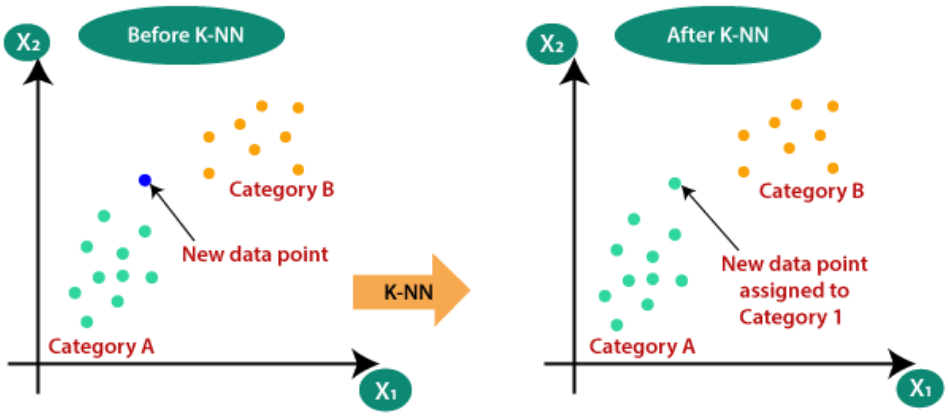

Figure 3 Illustration d'une classification par l'algorithme K-NN, cas de deux classe A et B.

1.8.4. Naive Bayes

Naive Bayes est un algorithme de classification probabiliste basé sur le théorème de Bayes. Son fonctionnement est simple : il calcule la probabilité qu'un nouvel exemple appartienne à chaque classe en utilisant les probabilités des classes et des caractéristiques. L'équation de Naive Bayes est :

$P(Ck|x)$=$P(Ck)\cdot P(x|Ck)P(x)$P(Ck|x)=P(x)P(Ck)·P(x|Ck)

Où :

P $(Ck|x)$ P (Ck|x) est la probabilité que l'exemple xx appartienne à la classe CkCk

P (Ck) P (Ck) est la probabilité a priori de la classe CkCk

P $(x|Ck)$ P (x|Ck) est la probabilité conditionnelle de l'exemple xx étant donné la classe CkCk

P (x) P(x) est la probabilité marginale de l'exemple xx.

Naive Bayes est rapide et efficace, mais son hypothèse d'indépendance conditionnelle peut être simpliste dans certaines situations. Malgré cela, il est largement utilisé dans des domaines comme la classification de textes et la détection de spam (5).

1.8.5. Les Machines à Vecteur Support (SVM)

Les SVMs, ou Machines à Vecteur Support, sont des outils de classification puissants en apprentissage automatique. Leur principe consiste à trouver un hyperplan dans un espace multidimensionnel pour séparer efficacement différentes classes de données en maximisant la marge entre les classes. Ils sont largement utilisés dans des domaines tels que la vision par ordinateur, la bio-informatique et la finance en raison de leur robustesse et de leurs performances élevées (6).

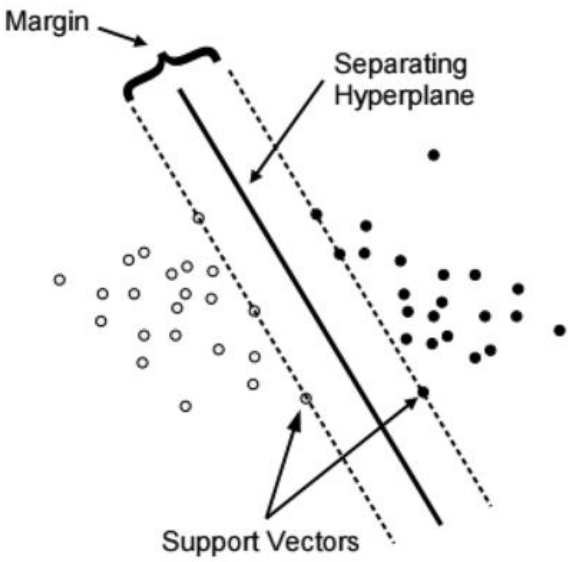

Figure 4 Classification linéaire par SVM.

La figure 4 présente un exemple de classification binaire. Les points aux extrémités des deux classes sont désignés comme des Vecteurs Supports, et c'est sur ces points que l'hyperplan est construit, se positionnant au milieu entre eux.

1.8.6. Les Réseaux de neurones

Les réseaux de neurones, aussi appelés réseaux de neurones artificiels (ANN) ou réseaux neuronaux simulés (SNN), constituent une composante essentielle de l'apprentissage en profondeur, un domaine de l'apprentissage automatique. Leur conception et leur fonctionnement sont inspirés du cerveau humain, reproduisant le processus par lequel les neurones biologiques échangent des signaux. Un réseau de neurones artificiels (ANN) est composé de couches de nœuds, incluant une couche d'entrée, une ou plusieurs couches cachées, et une couche de sortie. Chaque nœud, ou neurone artificiel, est connecté à d'autres et possède des poids et des seuils associés. Lorsque la sortie d'un nœud dépasse le seuil spécifié, ce nœud s'active et transmet des données à la couche suivante du réseau. Dans le cas contraire, aucune donnée n'est transmise à la couche suivante (7).

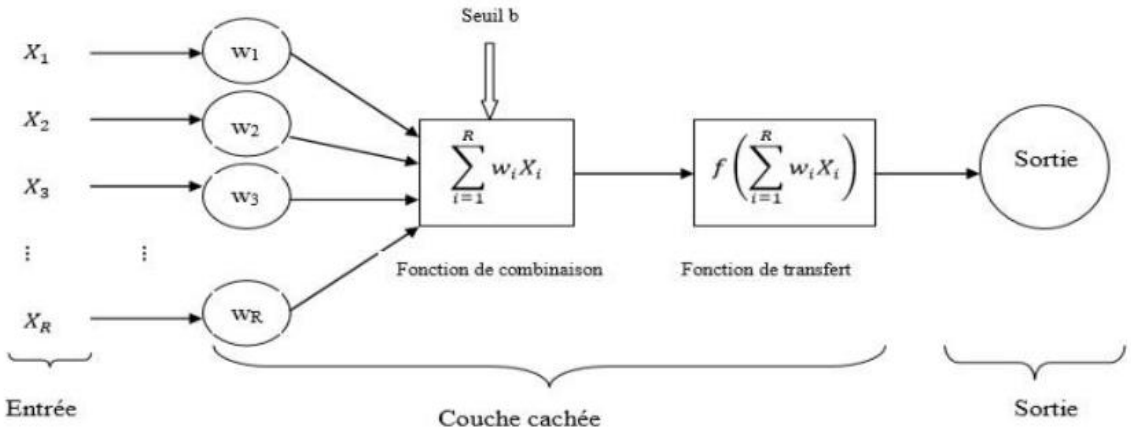

Figure 5 Conception d'un Réseau de Neurones. (Adapté de : Tufféry, 2012).

Les réseaux de neurones s'améliorent au fil du temps grâce à des données d'entraînement, renforçant ainsi leur précision. Une fois ces algorithmes d'apprentissage suffisamment perfectionnés, ils deviennent des outils informatiques et d'intelligence artificielle très efficaces, permettant une classification et une catégorisation rapides des données. Des tâches comme la reconnaissance vocale ou la reconnaissance d'images peuvent être accomplies en quelques minutes seulement, comparativement aux heures nécessaires pour une identification manuelle par des experts humains. Parmi les réseaux de neurones les plus connus, on trouve notamment l'algorithme de recherche de Google (8).

1.8.7. Bagging

Le bagging est un méta-algorithme faisant partie des méthodes ensemblistes : partant d'un algorithme de Machine Learning, il utilise de multiples fois cet algorithme pour obtenir un résultat plus fiable. Tout d'abord, le Bagging crée plusieurs sous-ensembles de données à partir de l'ensemble d'entraînement original en utilisant un échantillonnage aléatoire avec

remplacement. Ensuite, sur chaque sous-ensemble de données, un modèle de prédiction est entraîné indépendamment. Ces modèles peuvent être de différents types, tels que des arbres de décision, des SVM ou des réseaux de neurones. Une fois que tous les modèles individuels sont entraînés, leurs prédictions sont agrégées pour faire une prédiction finale. Cette agrégation peut se faire en utilisant un vote majoritaire pour les problèmes de classification ou en moyennant les prédictions pour les problèmes de régression (2).

Un inconvénient significatif de cette approche est sa propension à générer parfois une faible précision, en raison notamment des échantillons d'apprentissage de petite taille. Le Bagging, méthode qui vise à réduire le surapprentissage, se révèle efficace principalement lorsque le classificateur est instable face aux petits détails durant la phase d'apprentissage. Cela signifie qu'un léger changement dans les données peut engendrer une variation importante dans le modèle d'apprentissage produit. Un exemple d'algorithme sensible à ce phénomène est l'arbre de décision, notamment dans des espaces d'attributs à dimension très élevée, où la construction des nœuds supérieurs de l'arbre est particulièrement délicate.

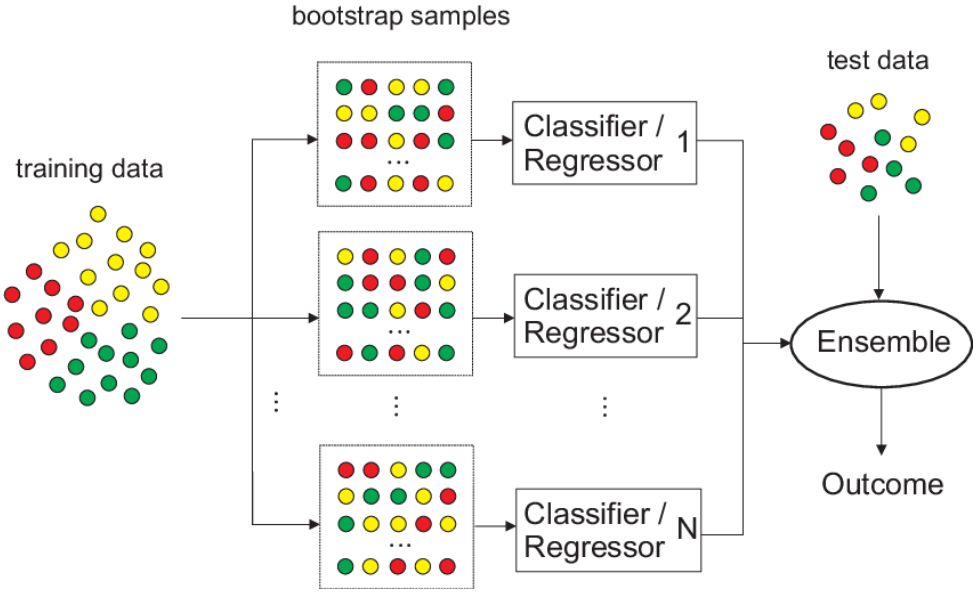

Figure 6 Schéma général de Bagging.

1.8.8. Boosting

Contrairement au Bagging, qui combine plusieurs modèles de manière parallèle, le Boosting construit une séquence de modèles de manière itérative, où chaque modèle est ajusté pour corriger les erreurs des modèles précédents. De cette manière, Boosting vise à améliorer la performance prédictive en se concentrant sur les exemples difficiles à classer.

L'approche d'entraînement varie selon le type de processus de boosting, également appelé algorithme de boosting. Cependant, la plupart des algorithmes de boosting suivent les étapes générales suivantes pour former le modèle (9) :

- Étape 1

L'algorithme de boosting initialise chaque échantillon de données avec un poids égal. Il alimente ensuite le premier modèle de base, généralement un algorithme simple, avec ces données. Ce modèle de base réalise des prédictions pour chaque échantillon.

- Étape 2

Le processus de boosting évalue les prédictions du modèle de base et augmente le poids des échantillons présentant des erreurs importantes. Les échantillons sont également pondérés en fonction de la performance du modèle. Ainsi, un modèle produisant des prédictions précises aura un poids plus important dans la décision finale.

- Étape 3

Les données pondérées sont ensuite transmises au modèle suivant, généralement un arbre de décision.

- Étape 4

Ces étapes 2 et 3 sont répétées jusqu'à ce que les erreurs d'entraînement soient inférieures à un seuil prédéfini.

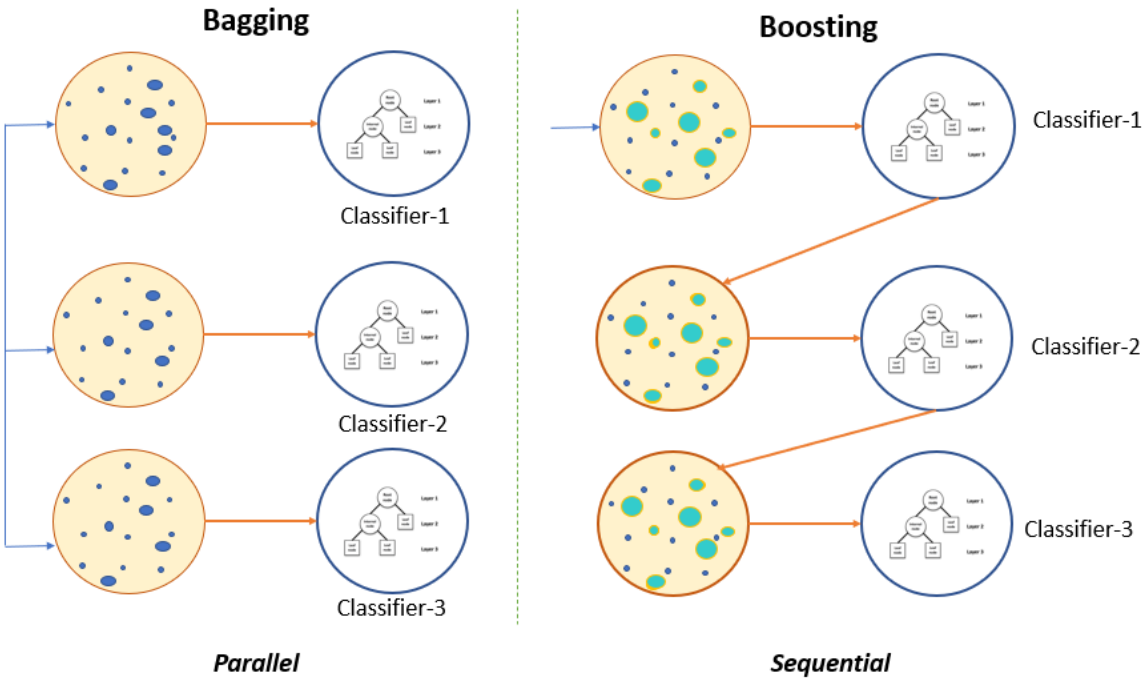

Figure 7 Comparaison entre Bagging et Boosting dans l'Ensemble Learning.

1.8.9. MLP (Multilayer Perceptron)

Les réseaux de neurones, et en particulier les perceptrons multicouches (MLP), sont utilisés pour découvrir des modèles cachés dans les données et pour la prise de décisions métier, comme

prévoir la demande de produits ou classer les clients. Le MLP utilise une architecture feedforward avec plusieurs couches masquées, ce qui en fait l'une des architectures les plus courantes (10).

Fonctionnalités principales du MLP :

- **Prévision et Classification** : Peut traiter des variables continues (prévoyant des valeurs proches de la réalité) ou catégorielles (classant les observations).
- **Redimensionnement** : Les covariables peuvent être normalisées avant l'entraînement.
- **Division des données** : Les données peuvent être divisées en ensembles d'apprentissage, de test et de rétention pour évaluer les performances.
- **Sélection de l'architecture** : Automatique ou manuelle, avec des options pour une ou deux couches masquées.
- **Fonctions d'activation** : Utilise des fonctions hyperboliques, sigmoïdes, identitaires ou softmax.
- **Méthodes d'apprentissage** : Inclut des options par lots, en ligne ou par mini-lots avec des algorithmes de descente de gradient.
- **Gestion des données manquantes** : Traite les valeurs manquantes des variables catégorielles comme valides.

Sortie :

La sortie inclut des tableaux croisés dynamiques, des graphiques variés (prévisions, résidus, courbes ROC, etc.), et des options pour enregistrer les prédictions et les pondérations.

En bref, le MLP est un réseau neuronal flexible et puissant, adapté pour des tâches de prévision et de classification avec diverses options de personnalisation et de sortie.

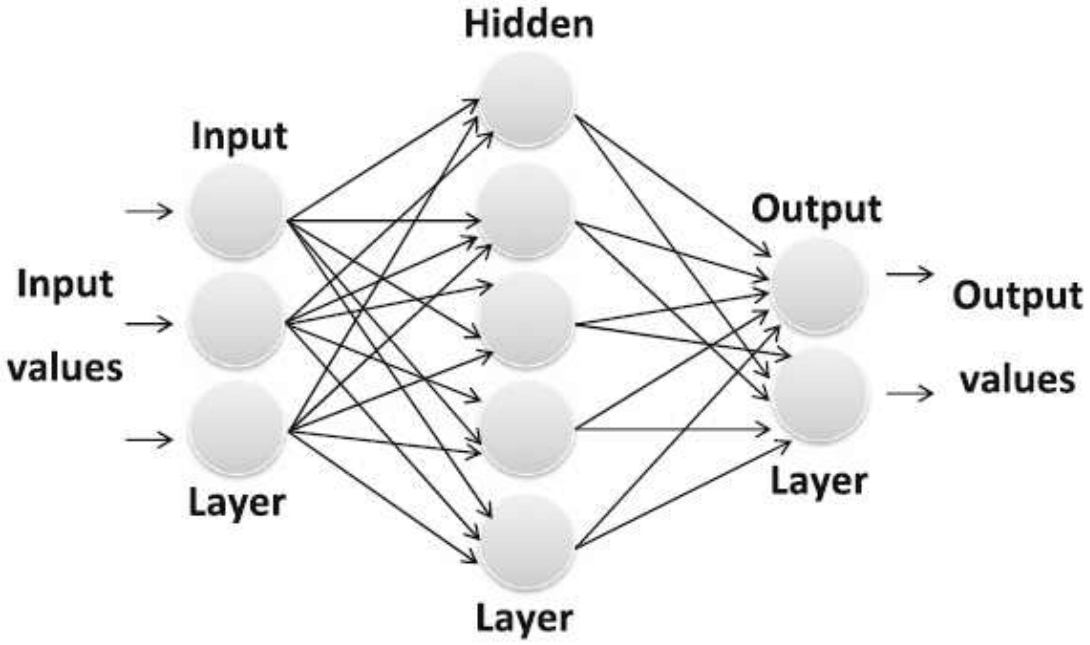

Figure 8 Structure de MLP.

1.9. Conclusion

En récapitulant ce chapitre sur la classification des textes, nous avons mis en lumière les différentes étapes du processus, de la représentation des données au prétraitement, en passant par les techniques de classification et leurs applications, ainsi que les algorithmes associés. En comprenant ces concepts fondamentaux et en explorant les applications de la classification des textes, nous sommes en mesure d'apprécier pleinement son importance dans divers domaines, de la catégorisation des documents à l'analyse des sentiments. En continuant à explorer et à maîtriser ces techniques, nous pouvons exploiter tout le potentiel des données textuelles pour des analyses plus approfondies et des décisions plus éclairées.

Chapitre 02

Racinisation

2.1. Introduction

Le traitement automatique du langage naturel (TALN) permet aux ordinateurs de comprendre le langage humain (texte et parole) en combinant l'intelligence artificielle et la linguistique. Cela leur permet d'effectuer des tâches telles que la traduction et de saisir le sens derrière nos mots. Les applications de TALN transforment le texte en vecteurs numériques, utilisant généralement des espaces de dimensions élevées pour la représentation des mots. Cela peut demander une mémoire substantielle. L'analyse morphologique regroupe les mots ayant des significations similaires, réduisant ainsi l'utilisation de la mémoire. La racinisation (stemming) est une méthode courante pour cela, alignant les mots apparentés sur le même vecteur.

La racinisation dans le traitement automatique du langage naturel (TALN) réduit les mots à leur forme de base ou racine, facilitant la normalisation du texte pour un traitement plus facile. Cette technique est cruciale dans des tâches telles que la classification de texte, la recherche d'information (IR) et le résumé de texte. Dans ce chapitre, nous explorerons le processus de racinisation et son application spécifique à la langue arabe. Nous commencerons par parler du raciniseurs (stemmers) les plus couramment utilisés (Porter, Lancaster…). Nous aborderons ensuite les caractéristiques de la langue arabe et les différentes approches de la racinisation en arabe. Enfin, nous présenterons les raciniseurs arabe de chaque approche (Isri, Light10, Khoja…) et leurs différentes règles.

2.2. Définition de la racinisation

La racinisation est un traitement lexical appliqué sur les mots selon leurs variations morphologiques. À savoir les flexions, dérivations et compositions. Donc il remplace, ou, regroupe un ensemble de mots de différentes morphologies qui expriment le même sens, par un seul terme qui est leurs racines (stem).

En d'autres termes, la racinisation est un processus informatique consistant à supprimer tous les suffixes et préfixes d'un mot pour générer le radical ou la racine. Par ailleurs, chaque langue a ses propres algorithmes de racinisation, c'est-à-dire que les règles appliquées en français ne sont pas les même pour une autre langue. Les algorithmes de racinisation sont basés sur des règles de troncatures de mots afin de produire des racines, en conséquence ils sont trop rapides (2).

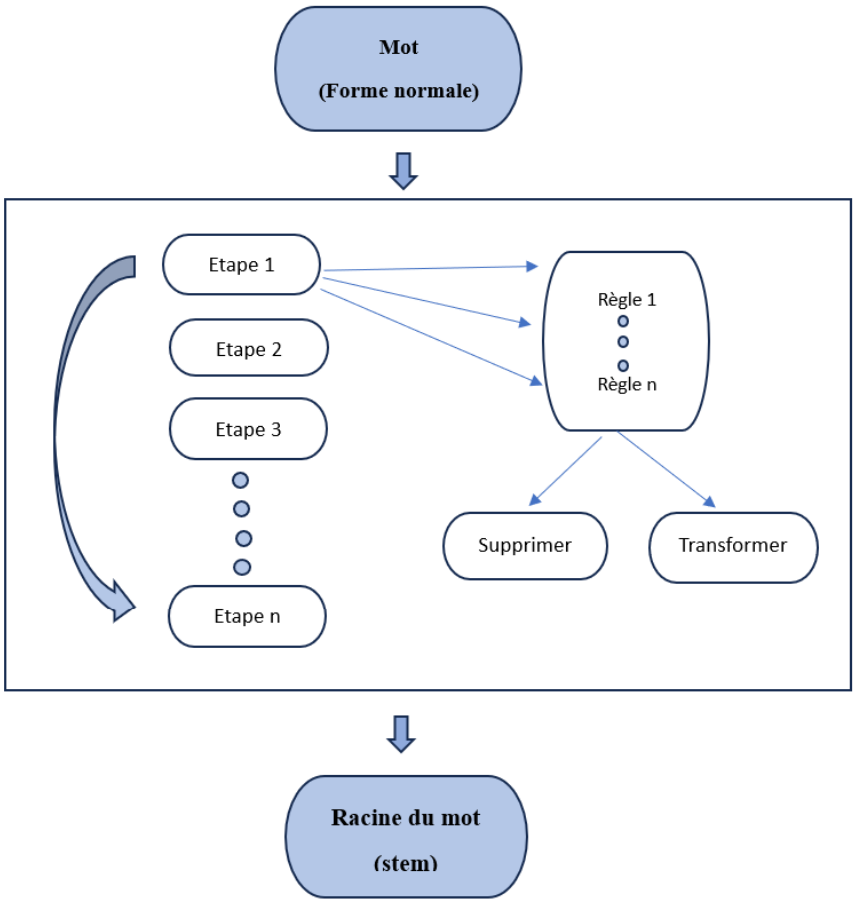

Figure 9 Le processus de la racinisation (stemming).

2.3. Les raciniseurs les plus couramment utilisés

2.3.1. Porter

L'algorithme Porter est l'un des algorithmes de racinisation les plus connus. Il a été créé par Porter en 1979 dans le laboratoire informatique de l'université du Cambridge (Angleterre). Il fait partie d'un grand projet dans le domaine de la recherche d'information. Il permet de retirer les affixes des mots (préfixes et suffixes) afin d'obtenir leur forme canonique. Cet algorithme est utilisé pour la langue anglaise, mais son efficacité devient très limitée lorsqu'il s'agit de traiter la langue française en raison de sa complexité morphologique plus importante. Néanmoins, il demeure un algorithme fondamental couramment enseigné en TALN.

- **Caractéristiques de porter**

L'algorithme PORTER comporte environ cinquante règles de racinisation/désuffixation réparties en sept phases successives comme le traitement du pluriel, des verbes à la troisième personne du singulier, du passé et du progressif. Chaque mot passe par toutes ces étapes et, si plusieurs règles pourraient leur être appliquées, c'est toujours celui qui a le suffixe le plus long qui est sélectionné. Dans la même étape, la racinisation/désuffixation est accompagnée de règles de recodage. Par exemple, le terme "troubling" deviendra "troubl" par enlèvement du suffixe marqueur du progressif -ing et sera ensuite transformé en "trouble" par application de la règle "bl" devient "ble". Cet algorithme comprend aussi cinq règles de contexte, qui indiquent les

conditions dans lesquelles un suffixe devra être supprimé. La terminaison en -ing, par exemple, ne sera enlevée que si le radical comporte au moins une voyelle. De cette manière, "troubling" deviendra "troubl", alors que "sing" restera "sing".

Le stemmer de Porter utilise une mesure, m, de la longueur d'un mot ou d'une partie de mot. Si C'est une séquence d'une ou plusieurs consonnes, et V une séquence d'une ou plusieurs voyelles, n'importe quelle partie de mot à la forme

$[C](VC)^m[V]$

Voici quelques exemples :

m=0 TR, EE, TREE, Y, BY.

m=1 TROUBLE, OATS, TREES, IVY.

m=2 TROUBLES, PRIVATE, OATEN, ORRERY.

Les règles de suppression d'un suffixe seront données sous la forme :

(Condition) S1→S2

Cela signifie que si un mot se termine par le suffixe S1 et que le radical avant S1 satisfait à la condition donnée, S1 est remplacé par S2. La condition est généralement donnée en termes de m, par ex.

(m > 1) EMENT →

Ici, S1 est « EMENT » et S2 est nul. Cela mapperait REPLACEMENT à REPLACE, puisque REPLACE est une partie de mot pour laquelle m = 2 (11).

```
Algorithme : PORTER
-Entrées : fichier contenant l'ensemble des mots
-Sorties : fichier contenant l'ensemble des mots stemmés
Début
    Tant que (non fin fichier) faire
        /* Étape 1 */
            ' Enlever et recoder forme pluriel
            ' Enlever « ed » et « ing » du verbe
        /* étape 2 */
            si (existe voyelle dans le stem) alors
                Transformer y en i
            Fin si
        /* étape 3 */
            'Indexer la lettre avant dernière
            ' Enlever le doublement s'il existe
        /* étape 4 */
            'Indexer la lettre finale du stem
            'Enlever la terminaison indiquée si possible
        /* étape 5 */
            si (stem a la forme <c>vcvc<v>) alors
                Enlever la terminaison
            Fin si
        /* étape 6 */
          si (plus qu'une séquence de consonne dans le stem) alors
                Enlever terminaison
          Fin si
        Retourner (stem)
        Fin tantque
Fin PORTER
```

Figure 10 Algorithme de PORTER.

2.3.2. Algorithme de Paice et Husk (Lancaster)

L'algorithme de Paice a été développé par Chris Paice à l'Université Lancaster dans les années 80. Il fait partie du groupe des raciniseurs (stemmers) algorithmiques. Il utilise un ensemble de règles pour extraire les racines, lesquelles sont stockées en dehors du code. Cet algorithme permet de traiter facilement une nouvelle langue en appliquant un ensemble différent de règles sans réécrire le code, à condition de faire quelques ajustements (pour chaque langue, il faut fournir la liste des voyelles acceptées et les règles de validité des racines). De cette manière, l'algorithme peut être facilement adapté à la gestion d'une nouvelle langue (12).

2.3.3. Algorithme de Krovetz

Bob Krovetz a créé le raciniseur Krovetz à l'université du Massachusetts en 1993. C'est un raciniseur léger, car il utilise la morphologie linguistique flexionnelle. Il s'agit d'un algorithme très complexe de faible force en raison des processus impliqués dans la morphologie linguistique et de sa nature flexionnelle.
Le « stemmer » Krovetz supprime efficacement et précisément les suffixes de manière suivante :

- La transformation d'un pluriel à sa forme singulière (par exemple '-ies', '-es', '-s').

Exemple : 'bunnies' devient 'bunny', 'boxes' devient 'box', 'cats' devient 'cat'.

- La transformation de passé au temps présent (par exemple '-ed').

Exemple : 'wanted' devient 'want'.

- L''élimination des '-ing' puis à travers un processus de vérification dans un dictionnaire pour tout recodage (aussi bien connaître des exceptions aux règles normales recodage), retourne le stem de mot.
Exemple : 'running' devient 'run' (12).

2.3.4. Algorithme de Snowball

Le raciniseur Snowball, également connu sous le nom de Porter2, est un algorithme de racinisation efficace conçu pour traiter et réduire les mots à leurs racines. Il a été développé par Martin Porter et est largement utilisé en raison de sa simplicité et de son efficacité. Le raciniseur Snowball prend en charge plusieurs langues et fournit des algorithmes spécifiques à chaque langue pour la racinisation.

2.4. Les caractéristiques de la langue arabe

L'arabe est une langue sémitique qui diffère syntactiquement, morphologiquement et sémantiquement des langues indo-européennes. Son système d'écriture comprend vingt-cinq consonnes et trois voyelles longues, écrites de droite à gauche et prenant différentes formes selon leur position dans le mot. De plus, l'arabe possède des voyelles courtes qui ne font pas partie de l'alphabet ; elles sont écrites sous forme de signes diacritiques au-dessus ou en dessous d'une consonne pour lui donner le son désiré, conférant ainsi au mot son sens voulu. Les textes sans voyelles sont considérés comme plus appropriés par la communauté arabophone, car ils sont largement utilisés dans les matériaux écrits et imprimés de tous les jours (livres, magazines, journaux, lettres, etc.). Cependant, dans le Saint Coran, les recueils imprimés de poésie classique, les manuels scolaires et certains dictionnaires papier arabes, les signes diacritiques apparaissent en entier. Généralement, dans les livres bien édités, certains textes imprimés et manuscrits, les signes diacritiques sont écrits partiellement ou de manière aléatoire là où les mots pourraient être ambigus ou difficiles à lire.

- 115 : Racines de deux caractères (et ces racines n'ont pas de dérivations à partir d'elles).
- 7198 : Racines de trois caractères.
- 3739 : Racines de quatre caractères.
- 295 : Racines de cinq caractères.

Ces racines se combinent avec divers schémas de voyelles pour former des noms et des verbes simples auxquels des affixes peuvent être attachés pour des dérivations plus complexes. Les schémas jouent un rôle important dans la lexicographie et la morphologie arabes. Par exemple, la racine "لعب" correspond au schéma "فعل", où d'autres lettres peuvent être ajoutées pour former un autre schéma. Par exemple, le schéma "مفعل" forme le mot "ملعب" en ajoutant la lettre "م" au morphème "لعب".

Les schémas du duel ou du pluriel sont composés en ajoutant des suffixes comme "ان", "ين" pour le duel et "ات", "ين", "ون" pour le pluriel. Il existe des formes irrégulières de schémas de pluriel appelées pluriels brisés. Dans ce cas, un nom au pluriel prend une autre forme morphologique différente de sa forme initiale au singulier (le tableau 1 montre les schémas singuliers et pluriels). Dans certains cas, un schéma pluriel peut avoir plus d'un schéma singulier, comme le montre le tableau(1) Par exemple, le mot "أطباء" a un singulier "طبيب" qui est comme "فعيل", mais le mot "ءعق" a un singulier "عاقل" qui est comme "فاعل".

Schéma pluriel	Schéma singulier
مفاعل	مفعل
مفاعيل	مفعول
أفعال	فعل
فعلاء	فعل، فعال، فاعل، فعيل
فعال	فاعل
أفعل	فعل
فواعل	فوعل، فاعل
أفعلة	فعيل، فعال

Tableau 1 Les schémas singuliers et pluriels.

Certaines des formes de pluriel brisées comme "فعلة", "فعالل", "فعائل" ont de nombreux schémas singuliers. Il n'existe pas de règles générales pour les couvrir toutes, et il est difficile de les traiter sans avoir un dictionnaire complet pour ces pluriels (13).

2.5. Les approches de la racinisation en arabe

2.5.1. La racinisation légère (Light-based stemming)

C'est le processus de suppression des affixes (préfixes, infixes et suffixes) pour obtenir les racines originales des mots sans nécessiter de trouver les racines. Des exemples de raciniseurs légers sont Light10 et Snowball et Tashapyne.

2.5.2. La racinisation basée sur la racine (Root-based stemming)

C'est le processus d'élimination des affixes et de modification de certaines lettres dans les mots pour obtenir le mot racine. Des exemples de raciniseurs lourds sont Khoja et ISRI.

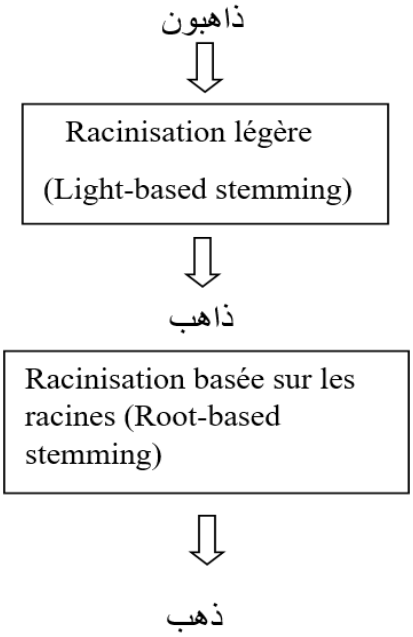

Figure 11 Exemple d'application d'un raciniseur léger et d'un raciniseur basé sur les racines.

2.5.3. La racinisation statistique (Statistical stemming)

L'approche statistique de la racinisation en arabe s'éloigne des règles fabriquées à la main. Elle analyse un grand corpus de textes arabes pour trouver les préfixes et suffixes fréquemment présents, qui sont susceptibles d'être des affixes supprimables. Des techniques telles que les n-grammes (séquences de lettres) et la co-occurrence (la manière dont les mots apparaissent ensemble) aident à identifier ces affixes. Cette approche peut s'adapter à des types de texte spécifiques et être potentiellement plus précise, mais elle peut être coûteuse en termes de calcul et ne gère pas toujours parfaitement les ambiguïtés (14).

2.5.4. La racinisation par l'intelligence artificielle

Inspirés par la large applicabilité des réseaux de neurones (NN), Alserhan et Ayesh ont proposé l'utilisation d'un réseau de neurones à rétropropagation (BPNN) pour la racinisation du texte arabe. Leur approche était motivée par les limitations des méthodes actuelles basées sur les racines, qui reposent sur des règles morphologiques étendues, impactant le stockage et le temps de traitement. Ils ont classé les lettres arabes en quatre catégories en fonction de leur fréquence, représentant les affixes arabes, et ont attribué à chaque catégorie une valeur binaire de trois chiffres. Pour l'entraînement, ils ont utilisé un ensemble de données de 250 mots avec leurs racines correctes. L'étude a démontré que cette approche par réseau de neurones pouvait atteindre une précision de 84 % dans l'identification des racines correctes. L'avantage principal de leur méthode est qu'elle ne dépend pas des règles morphologiques. Cependant, elle présente deux inconvénients majeurs : elle ne traite que les mots de quatre caractères maximums, et les résultats ont été testés sur un ensemble de données relativement petit de 1000 mots (14).

2.6. Les raciniseurs basé sur les racines (Root-based stemmers)

2.6.1. Khoja

Le Khoja Stemmer a été développé par M. M. Khoja et A. P. Dawod en 1999. Khoja supprime les suffixes, les infixes et les préfixes. Il utilise la correspondance de motifs pour extraire les racines des mots ; cependant, ce travail présente certaines lacunes et inconvénients, notamment avec certains mots représentant des 'pluriels brisés'. Par la suite, Taghva et al. (2005) ont amélioré l'algorithme de (Khoja & Garside 1999) en éliminant le besoin d'utiliser un dictionnaire pour extraire la racine. Tandis que dans Al-Kabi et al. (2015), le raciniseur a été amélioré en ajoutant des règles et des motifs supplémentaires. Récemment, des techniques d'extraction de racine basées sur des motifs ont été développées sans utiliser de dictionnaire (Al-Kabi et al. 2015, Nehar et al. 2016) (15).

Les règles du stemmer khoja sont les suivantes (13) :

1. Supprimer le diacritique représentant la vocalisation.

2. Supprimer les mots vides, la ponctuation et les chiffres.

3. Supprimer l'article défini "ال".

4. Supprimer la conjonction inséparable "و".

5. Supprimer les suffixes.

6. Supprimer les préfixes.

7. Mettre en correspondance le résultat avec une liste de motifs. Si une correspondance est trouvée, extraire les caractères dans le motif représentant la racine.

8. Mettre en correspondance la racine extraite avec une liste de racines "valides" connues.

9. Remplacer les lettres faibles "أ,ى,ئ" par "ا"

10. Remplacer toutes les occurrences de hamza "أ,إ,ؤ" par "ء"

11. Vérifier si les racines à deux lettres doivent contenir un double caractère. Si c'est le cas, le caractère est ajouté à la racine.

Voici un exemple de la racinisation de khoja :

Mot original	Racine extraite
مكتبة	كتب
مدارس	درس
استملاك	ملك
قرءان	ق ر أ

Tableau 2 Exemples de la racinisation de khoja.

2.6.2. ISRI

En 2005, Kazem T et al ont proposé un algorithme pour la racinisation arabe qui partage de nombreuses caractéristiques avec le stemmer Khoja. Le raciniseur de l'Institut de recherche en sciences de l'information (ISRI) utilise une approche similaire à celle du raciniseur Khoja pour la racinisation des mots, mais n'emploie pas de dictionnaire de racines pour la recherche. De plus, si un mot ne peut pas être racinisé, ISRI normalise le mot et renvoie une forme normalisée (par exemple, en supprimant certains déterminants et motifs de fin) au lieu de laisser le mot inchangé (15).

Il définit des ensembles de signes diacritiques et de classes d'affixes comme indiqué dans le Tableau 4.

Signe	Description	Exemples
D	Diacritiques	بْ, بِ, بً, بِ, بُ,بَ
P1	Préfixes de longueur un.	ب، ف، ك، ل، س، ت
P2	Préfixes de longueur deux.	ال، لل
P3	Préfixes de longueur trois.	ولل، وال، كال، بال
S1	Suffixes de longueur un.	ة، ه، ي، ا، ن،ت
S2	Suffixes de longueur deux.	ون، ات، ان، ين، تن، كم، هن
S3	Suffixes de longueur trois.	تمل، همل، تان، تين

Tableau 3 Ensembles d'affixes.

Les règles De ISRI sont les suivantes (13) :

1. Supprimer les signes diacritiques représentant les voyelles.

2. Normaliser le hamza qui apparaît sous plusieurs formes distinctes en combinaison avec diverses lettres, pour le ramener à une seule forme "أ".

3. Supprimer respectivement les préfixes de longueur trois et de longueur deux.

4. Supprimer le connecteur "و" s'il précède un mot commençant par "و".

5. Normaliser "إ,أ,آ" en "ا".

6. Renvoyer la racine si elle est inférieure ou égale à trois.

7. Considérer quatre cas en fonction de la longueur du mot, chacun avec des règles spécifiques pour la racinisation, y compris l'extraction de racines pertinentes basées sur des motifs prédéfinis et la suppression des suffixes et préfixes.

Voici un exemple de la racinisation d'ISRI :

Mot original	Racine extraite
وفاقية	وف ق
وضوء	وض ء
متعاونين	ع ون
إيمان	ا م ن
مقتنعون	ق ن ع
آخر	ا خ ر
مستفيدين	ف ي د

Tableau 4 Exemples de la racinisation de ISRI.

2.7. Les raciniseurs légère (Light-based stemmers)

2.7.1. Light10

Larkey et al. Ont développé un ensemble de stemmers légers. Le plus largement utilisé dans cet ensemble est le light 10, et chaque stemmer se distingue des autres par le nombre total de préfixes et de suffixes à supprimer. En fait, les préfixes et suffixes ont été accumulés au light 10. Dans le light-10, Larkey suggère de supprimer légèrement les préfixes (ال، و ال، بال، كال، فال) et les suffixes (لل، و ها، ان، ات، ون، ين، يه، ية، هـ، ة، ي). Cependant, afin d'éviter la suppression des affixes qui font partie des racines des mots (et qui doivent donc être conservés), l'algorithme est traité avec trois règles (14) :

1. Supprimez "و" de light2, light3, et light8 ainsi que light10 si le reste du mot compte 3 caractères ou plus.

2. Supprimez tout article défini si cela laisse 2 caractères ou plus.

3. Passez en revue la liste des suffixes une fois dans l'ordre de droite à gauche (Tableau 4).

	Supprimer les préfixes	Supprimer les suffixes
Light1	ال، وال، بال، كال، فال	Aucun
Light2	ال، وال، بال، كال، فال، و	Aucun
Light3	،،	ه، ة
Light8	،،	ها، ان، ات، ون، ين، يه، ية، هـ، ة، ي
Light10	ال، لل، وال، بال، كال، فال، و	،،

Tableau 5 Liste des affixes supprimées par light1, Light2, Light3, Light8, Light10.

Voici un exemple de la racinisation de Light10 :

Mot original	Racine extraite
والكتابة	كتاب
كلمات	كلم
واستكشافاتهم	كشف
إجراءات	جراء

Tableau 6 Exemple de la racinisation de Light10.

2.7.2. CondLight

Al-Lahham et al. (2018) ont proposé un raciniseur légèr conditionnel, appelé Condition Light, ou en abrégé 'CondLight', comme une amélioration de la méthode Light10. Cette amélioration consiste à ajouter un ensemble de nouveaux affixes à éliminer s'ils satisfont à une ou plusieurs des conditions proposées. Ces conditions sont dérivées de la nature morphologique de la langue arabe.

La liste des affixes utilisés par CondLight, comparée à la liste utilisée par les variantes du stemmer Light, est présentée dans le Tableau 5 (16).

	Supprimer les préfixes	Supprimer les suffixes
Light1	ال، وال، بال، كال، فال	Aucun
Light2	ال، وال، بال، كال، فال، و	Aucun
Light3	،،	ه،ة
Light8	،،	ها، ان، ات، ون، ين، يه، ية، ه، ة، ي
Light10	ال، لل، وال، بال، كال، فال، و	،،
Condlight	ال، لل، وال، بال، كال، فال، و ب، ت، ل، ي، ف، س.	ها، ان، ات، ون، ين، يه، ية، ه، ة، ي هم، هن، وا.

Tableau 7 Liste des affixes ajouté par Condlight.

Voici un exemple de la racinisation de Condlight et Light10 aux mêmes mots :

Mot original	Light10	Condlight
يدرسها	يدرس	درس
تطورت	تطورت	طور
لمهرجانات	لمهرجان	مهرجان
تعاملت	تعامل	عامل

Tableau 8 Exemple d'application de Light10 et CondLight aux mêmes mots.

2.7.3. Tashaphyne

Zerrouki (2010) a présenté Tashaphyne, c'est est un raciniseur léger et un segmenteur arabe. Il prend principalement en charge la racinisation légère (suppression des préfixes et des suffixes) et donne toutes les segmentations possibles. Il utilise un automate fini modifié qui permet de générer toutes les segmentations. Il offre la racinisation et l'extraction de racine en même temps contrairement aux autre raciniseurs(stemmers).

2.8. Conclusion

Le but de la racinisation (stemming) est de simplifier et de normaliser les mots, ce qui aide à améliorer les performances de la recherche d'informations, de la classification de texte et d'autres tâches de TALN. Dans ce chapitre, nous avons exploré le processus de racinisation et son application à la langue arabe. Nous avons présenté quelques raciniseurs célèbres (Porter, Lovins…). Ensuite, les caractéristiques distinctives de la langue arabe et les diverses approches de la racinisation de la langue.

Chapitre 03

Méthodologie de travail

3.1. Introduction

Dans les deux chapitres précédents, nous avons examiné de près la catégorisation des textes, en explorant les diverses méthodes d'algorithmes de classification. Nous avons analysé les algorithmes tels que les machines à vecteurs de support (SVM), les Naive Bayes, et les réseaux de neurones. De plus, nous avons étudié en détail le processus de racinisation (stemming), crucial pour préparer les données textuelles en vue de la classification. Nous avons examiné différentes techniques de racinisation, telles que l'algorithme de Porter et la racinisation en langue arabe, en mettant en évidence leurs impacts sur la réduction de la variance et la précision de la classification.

Ce chapitre présente la méthodologie de travail de notre approche de classification de texte, axée sur l'exploration et la comparaison des performances de divers algorithmes de classification, ainsi que l'effet de différentes techniques de prétraitement sur deux collections textuelles distinctes. Notre démarche méthodologique repose sur une série d'étapes rigoureuses, allant du prétraitement initial des données à l'évaluation comparative des résultats obtenus.

3.2. Méthodologie de travail

3.2.1. Collections textuelles

- **La première collection textuelle « Arabic Wikipedia Corpus »**

Les informations sur le corpus d'Arabic Wikipédia, compilées par Motaz Saad du Département d'informatique de la Faculté de technologie de l'information de l'Université islamique de Gaza, sont présentées dans le Tableau 9. Ce corpus a débuté en 2003 avec 655 articles et a progressé pour contenir 459 208 articles jusqu'au 20 janvier 2017. Les données comprennent le nombre de documents dans le corpus (|d|), le nombre de mots dans le corpus (|w|), et la taille du vocabulaire (|v|), représentant le nombre de mots uniques (17).

|d|	459,208 documents
|w|	83.5M mots
|v|	4.7M mots uniques

Tableau 9 Arabic Wikipedia Corpus

Le corpus est disponible en ligne sur https://github.com/motazsaad/arWikiExtracts pour la communauté de recherche. Le Tableau 10 montre les mots les plus fréquents dans le corpus d'Arabic Wikipedia, tandis que le Tableau 11 montre les mots les plus fréquents en excluant les mots vides. Les mots les plus fréquents représentent les mots courants dans une langue. Si nous considérons le corpus d'Arabic Wikipedia comme une représentation de la langue arabe, alors ces listes représentent les mots les plus fréquents dans la langue arabe.

Mots en arabes	Fréquence
في	912670
من	664693
على	313382
إلى	246869
أن	160484
عام	122708
التي	122642
عن	110351
أو	100635
مع	92920
كان	89223
بن	83028
الذي	76458
بعد	75980
و	72981
هذه	71759
ما	70403
هو	66420
حيث	65653
بين	64536
هذا	64420
هي	59250
ذلك	59228
كما	57971
وقد	56778
كانت	53043
وفي	51493
لا	50895
سنة	47149
وهو	42505

Tableau 10 les mots les plus fréquents dans le corpus Arabic Wikipedia.

Mots en arabe	Fréquence
عام	122708
سنة	47149
الله	37305
عبد	35873
مدينة	35259
عدد	32556
محمد	28823
المدينة	26614
العالم	25478
المتحدة	23319
العربية	22606
أول	22458
القرن	22233
أخرى	21754
علي	21481
أحد	20706
الأول	20463
العديد	19791
نادي	19646
عشر	19107
منطقة	19077
فقد	18935
الأولى	18148
العام	17820
ابن	17612
مصر	17514
الثاني	16684
إحدى	15800
تقع	15686
حوالي	15602

Tableau 11 les mots les plus fréquents, à l'exclusion des mots vides.

- **Création des catégories dans la collection textuelle**

Cette collection textuelle est considérable, contenant un nombre important de documents, comme nous l'avons déjà mentionné. Nous avons choisi de travailler avec un document spécifique nommé "AA" et avons sélectionné neuf fichiers parmi ceux qu'il contient. Ensemble, ces neuf fichiers contiennent un total de 500 textes arabes. Chaque texte est associé à un titre et à un identifiant unique, cependant, aucun d'entre eux n'est préalablement catégorisé. Pour faciliter l'analyse, nous avons créé une liste de onze catégories pertinentes :

1) Biographie
2) Science
3) Économie
4) Technologie
5) Géographie

6) Histoire
7) Religion
8) Astronomie
9) Linguistique
10) Politique
11) Biologie

Nous avons ensuite attribué à chaque texte sa catégorie appropriée, en l'indiquant à côté de son titre et de son identifiant. Cette approche nous a permet d'organiser et de structurer les textes pour une analyse plus approfondie et cohérente.

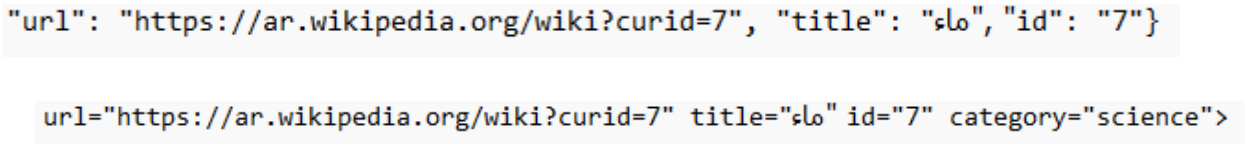

Figure 12 Comparaison visuelle de la base de données avant et après l'ajout de la catégorie devant chaque texte.

- **La deuxième collection textuelle (Arabic Poem Classification Challenge dataset)**

Le Jeu de Données du Défi de Classification de Poèmes Arabes, tiré du site Kaggle, comprend une collection de poèmes arabes avec les caractéristiques suivantes (18) :

Titre : Cette colonne contient le titre de chaque poème, offrant une brève description ou un nom reflétant souvent le thème ou le sujet du poème.

Auteur : La colonne de l'auteur indique le nom du poète ayant écrit le poème respectif, attribuant ainsi la paternité aux esprits créatifs derrière les œuvres poétiques.

Ère : La colonne de l'ère représente la période historique à laquelle chaque poème appartient, offrant des insights sur le contexte culturel, social et littéraire de l'époque.

Poème : Cette colonne contient le texte réel du poème, comprenant les vers, les strophes ou les lignes composant la composition poétique. Cette colonne constitue le principal objet d'analyse et de classification dans ce défi.

Le jeu de données vise à proposer une sélection diversifiée de poèmes arabes provenant de différentes périodes historiques, permettant ainsi aux participants d'explorer et d'analyser la poésie de diverses époques. L'objectif du défi est de développer des modèles de classification capables de catégoriser précisément les poèmes en fonction de leur époque, démontrant ainsi la capacité à comprendre et à identifier les caractéristiques distinctives associées à chaque période.

Fichiers	Deux fichiers
Taille	45,25 Mo
Type	csv

Tableau 12 Arabic Poem Classification Challenge dataset

Cette collection textuelle contient deux fichiers : le premier nommé poems.csv et le second test.csv, nous avons pris uniquement le fichier poems.csv et avons travaillé avec.

3.2.2. Prétraitement

Le prétraitement de texte est une étape essentielle dans le traitement du langage naturel (NLP). Il vise à nettoyer et à préparer les données textuelles avant leur analyse ou leur utilisation dans des modèles. Voici les étapes que nous avons suivies dans notre processus de prétraitement :

✓ **Importation des données**

1. Tout d'abord, nous avons stocké notre base de données dans notre Google Drive.
2. Ensuite, nous avons monté notre Google Drive dans notre environnement Google Colab en utilisant la commande drive.mount('/content/drive').
3. Après avoir monté notre Google Drive, nous avons accédé au chemin où se trouvent nos fichiers en utilisant le chemin relatif dans notre Google Drive.
4. Nous avons utilisé la bibliothèque Python os pour lister les fichiers dans ce répertoire à l'aide de os.listdir(folder_path).

✓ **Suppression de la ponctuation, des chiffres et des caractères spéciaux**

Nous avons utilisé l'expression régulière re.sub() pour remplacer tous les caractères qui ne sont pas des lettres ou des espaces par une chaîne vide ainsi pour effectuer les opérations suivantes :

- Suppression les chiffres occidentaux (0-9) en utilisant le motif r'[0-9]'.
- Suppression les chiffres arabo-indiens (٠ - ٩) avec le motif r'[\u0660-\u0669]'.
- Suppression les chiffres persans (۰ - ۹) à l'aide du motif r'[\u06F0-\u06F9]'.
- Suppression les caractères antislash (\) en utilisant le motif r'\\.
- Suppression les caractères non arabes et les caractères spéciaux en utilisant le motif r'[^\u0600-\u06FF\s]'. Ce motif conserve uniquement les caractères arabes (dans la plage Unicode \u0600-\u06FF) et les espaces, supprimant ainsi tous les autres caractères.

✓ **Tokenisation**

Utilisation de word_tokenize de NLTK pour diviser le texte en mots.

✓ **Suppression des mots vides**

Chargement des mots vides arabes à l'aide de stopwords.words('arabic').

Parcourir chaque mot et le conserver s'il n'est pas dans la liste des mots vides.

- ✓ **Racinisation (Stemming)**

Nous avons utilisé trois racinisateurs différents (ISRI Stemmer, TASHAPHYNE Stemmer, SNOWBALL Stemmer) dans notre étude pour raciner chaque mot.

- ✓ **Réassemblage du texte nettoyé**

Nous avons utilisé la fonction join() pour assembler les mots racinisés en une seule chaîne pour former un nouveau texte nettoyé.

- ✓ **Nettoyage et organisation des données**

Dans notre étude, nous avons utilisé une boucle for pour parcourir une liste de fichiers dans un répertoire spécifique. Pour chaque fichier, nous avons extrait les documents et leurs catégories associées, puis nous avons appliqué la fonction clean_arabic_text() pour nettoyer chaque document. Les documents nettoyés ont été ajoutés à une liste appelée my_list, tandis que les catégories correspondantes ont été ajoutées à une autre liste appelée my_cats.

3.2.3. La représentation des textes

Après avoir effectué le prétraitement des données, nous avons utilisé la méthode CountVectorizer de la bibliothèque scikit-learn pour construire un vecteur de caractéristiques à partir de notre corpus de texte. Cette approche, connue sous le nom de sac de mots (bag of words), consiste à représenter chaque document par un vecteur où chaque élément correspond à la fréquence d'apparition d'un terme spécifique dans ce document. Ensuite, en combinant ces vecteurs, nous avons obtenu la matrice terme-document, qui représente l'ensemble du corpus en capturant la fréquence des termes dans chaque document. Cette matrice est ensuite utilisée comme entrée pour appliquer des algorithmes de classification de texte.

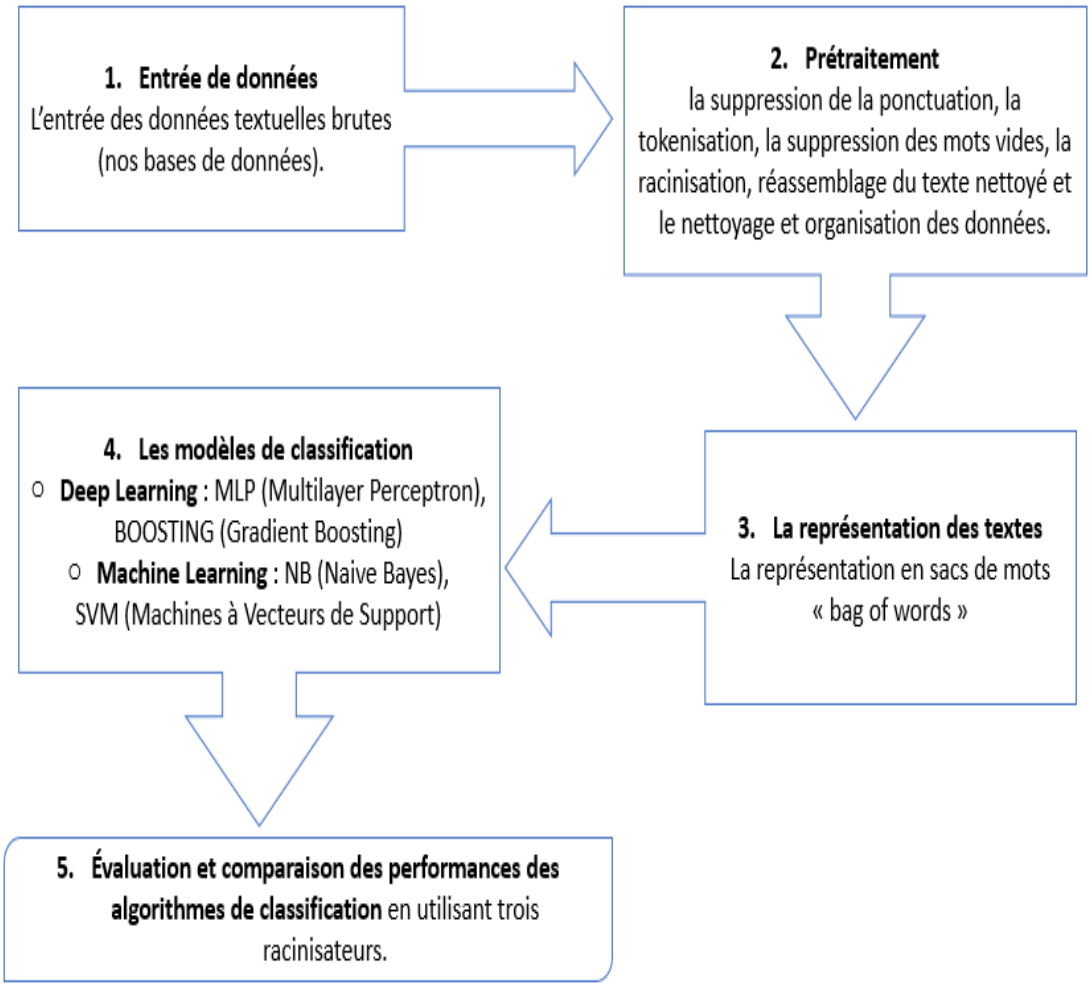

Figure 13 schéma englobe notre processus de classification.

3.2.4. Évaluation des performances des algorithmes de classification de texte

Après avoir divisé nos données en ensembles d'entraînement et de test, nous avons appliqué différents algorithmes de classification à nos données d'entraînement. Pour évaluer les performances de chaque algorithme, nous avons utilisé plusieurs mesures, notamment la précision (precision), le rappel (recall) et le score F1 (f1-score). Ces mesures sont calculées pour chaque classe individuellement, ainsi que pour l'ensemble des données (accuracy), ce qui nous permet d'obtenir une évaluation complète des performances de chaque algorithme de classification.

- **Le rappel (recall)**

C'est le nombre de positifs bien prédit (Vrai Positif) divisé par l'ensemble des positifs (Vrai Positif + Faux Négatif).

$$Rappel = \frac{nombre\ de\ vrais\ positifs}{(nombre\ de\ vrais\ positifs + nombre\ de\ faux\ négatifs)}$$

- **La précision**

C'est le nombre de positifs bien prédit (Vrai Positif) divisé par l'ensemble des positifs prédit (Vrai Positif + Faux Positif).

$$Précision = \frac{nombre\ de\ vrais\ positifs}{(nombre\ de\ vrais\ positifs + nombre\ de\ faux\ positifs)}$$

- **Le score F1 (f1-score)**

Le F1-score est une mesure de performance utilisée en classification qui combine à la fois la précision et le recall. Il est calculé comme la moyenne harmonique de la précision et du recall.

$$F1 - score = 2 \times \frac{précision \times recall}{précision + recall}$$

- **L'exactitude (accuracy)**

Mesure la proportion d'observations correctement classées parmi toutes les observations. La formule de l'accuracy est la suivante :

$$Accuracy = \frac{vrais\ positifs + vrais\ négatifs}{nombre\ total\ d'observations}$$

3.3. Conclusion

Dans ce chapitre, nous avons présenté en détail notre méthodologie pour la classification de textes arabes, en utilisant deux bases de données distinctes : le corpus d'Arabic Wikipedia et le jeu de données du Défi de Classification de Poèmes Arabes. Nous avons entamé notre démarche par le prétraitement des données pour nettoyer et préparer les textes avant l'analyse. Les étapes comprenaient la suppression de la ponctuation, la tokenisation, l'élimination des mots vides et la racinisation avec trois différents stemmers.

Par la suite, nous avons représenté les textes en utilisant la méthode CountVectorizer, permettant de créer une matrice terme-document essentielle pour appliquer les algorithmes de classification. Nous avons mis en œuvre plusieurs algorithmes de classification et avons évalué leurs performances à l'aide de métriques telles que la précision, le rappel, le score F1 et l'exactitude.

Les résultats de nos algorithmes de classification, ainsi que l'analyse des performances en fonction des prétraitements appliqués, seront abordés dans le chapitre suivant. Ce chapitre se concentre donc sur la méthodologie rigoureuse et les préparations nécessaires pour une classification efficace des textes arabes.

Chapitre 04

Réalisation et résultats

4.1. Introduction

Dans ce quatrième chapitre, nous allons présenter les résultats de notre étude sur la classification de textes, en mettant en lumière les différents aspects méthodologiques et analytiques. Notre approche a consisté à explorer l'efficacité de quatre algorithmes de classification, regroupant des techniques de machine learning et de deep learning. Nous avons appliqué ces algorithmes à trois racinisateurs (stemmers) et à deux ensembles de données distincts, afin de mesurer leur performance dans des contextes variés. En outre, nous avons comparé les performances des algorithmes avec et sans l'utilisation de racinisateurs pour déterminer l'impact de ces derniers sur les résultats. Les performances des modèles seront évaluées à l'aide de plusieurs métriques, notamment la précision, le rappel, le score F1 et l'accuracy. Ces mesures nous permettront de réaliser une évaluation complète et approfondie de l'efficacité des modèles de classification dans notre contexte spécifique. Nous discuterons également des implications de nos résultats et des perspectives d'amélioration futures pour la classification de textes.

4.2. Outils et langages utilisés

4.2.1. Python

Le langage Python est un langage de programmation open source, multiplateforme et orienté objet. Grâce à ses bibliothèques spécialisées, il est utilisé dans de nombreux domaines, tels que le développement de logiciels, l'analyse de données et la gestion d'infrastructures. Contrairement à HTML, qui est spécifiquement destiné à la programmation web, Python offre une grande polyvalence (19).

En tant que langage interprété, Python permet d'exécuter du code sur n'importe quel ordinateur. Il est adapté aussi bien aux débutants qu'aux experts, permettant de créer des programmes de manière simple et rapide.

4.2.2. Anaconda

Anaconda est un outil libre et open source destiné à la programmation en Python et R. Il est largement utilisé dans les domaines de la science des données, de l'intelligence artificielle et de la machine learning. Cette distribution scientifique de Python inclut de nombreux packages nécessaires à l'analyse de données et sert également de gestionnaire d'environnement open source (20).

Avec plus de 20 millions d'utilisateurs dans le monde, Anaconda comprend notamment :

- ✓ Une installation de l'environnement Python,
- ✓ des IDE de dernière génération comme Jupyter et Spyder,
- ✓ des packages de Data Science tels que Pandas, Numpy et Scikit-Learn,
- ✓ l'outil Conda pour la gestion des environnements et des packages.

Grâce à ses divers outils et à l'environnement Python, Anaconda facilite la collecte et la transformation des données à grande échelle. Jupyter, l'un des IDE inclus, prend désormais en charge plus de 40 langages de programmation.

4.2.3. Jupyter notebook

Un Jupyter Notebook est une application web open source permettant aux data scientists de créer et de partager des documents contenant du code en direct, des équations et des ressources multimédia. Ils sont particulièrement utiles pour présenter le travail de l'équipe de données en combinant code, texte en markdown, liens et images, et peuvent être exécutés cellule par cellule pour une meilleure compréhension du code (21).

Les Jupyter Notebooks peuvent être convertis en divers formats de sortie standard (HTML, PowerPoint, LaTeX, PDF, ReStructuredText, Markdown, Python) via l'interface web, ce qui facilite le partage du travail. Ils comportent deux composants : une interface web pour saisir du code ou du texte et un noyau back-end qui exécute le code et retourne les résultats.

4.3. Bibliothèques utilisées

4.3.1. NLTK (Natural Language Toolkit)

NLTK est une plateforme de premier plan pour créer des programmes en Python permettant de travailler avec des données de la langue humaine. Elle offre des interfaces faciles à utiliser pour plus de 50 corpus et ressources lexicales, telles que WordNet, et inclut une suite de bibliothèques pour diverses tâches de traitement de texte : classification, tokenisation, stemming, tagging, parsing et raisonnement sémantique. NLTK propose également des wrappers pour des bibliothèques de traitement du langage naturel de qualité industrielle et dispose d'un forum de discussion actif.

Grâce à un guide pratique qui introduit les bases de la programmation en parallèle avec des sujets en linguistique computationnelle, ainsi qu'une documentation API complète, NLTK est adapté aux linguistes, ingénieurs, étudiants, enseignants, chercheurs et utilisateurs industriels. NLTK est disponible pour Windows, Mac OS X et Linux. De plus, NLTK est un projet gratuit, open source et communautaire (22).

4.3.2. Scikit-Learn

Scikit-Learn est une bibliothèque Python qui propose des implémentations efficaces de nombreux algorithmes courants, avec une API claire et cohérente. L'un de ses principaux avantages est qu'une fois que vous avez compris l'utilisation de base et la syntaxe pour un type de modèle, il est très facile de passer à un autre modèle ou algorithme. En plus de la modélisation, la bibliothèque prend en charge les étapes de prétraitement, comme nous le verrons dans le reste de cet article (23).

4.3.3. Pandas

La bibliothèque open source Pandas est spécialement conçue pour la manipulation et l'analyse de données en Python. Elle est performante, flexible et simple d'utilisation. Grâce à Pandas,

Python permet de charger, d'aligner, de manipuler et de fusionner des données. Les performances sont particulièrement impressionnantes lorsque le code source back-end est écrit en C ou en Python (24).

Le nom « Pandas » est une contraction de « Panel Data », qui désigne des ensembles de données incluant des observations sur plusieurs périodes temporelles. Cette bibliothèque a été créée comme un outil de haut niveau pour l'analyse de données en Python.

4.3.4. OS

Le module os en Python est une bibliothèque standard qui permet de travailler avec le système d'exploitation. Il offre un ensemble de fonctions pour effectuer des opérations de bas niveau, telles que la création de processus, l'allocation de mémoire et la gestion des signaux.

Grâce aux fonctions fournies par le module os, il est possible d'interagir avec des fichiers, des répertoires, des chemins, des processus et des utilisateurs. De plus, le module os propose des méthodes pour accéder à l'environnement système, obtenir des informations sur le système d'exploitation, et bien plus encore (25).

4.4. Collections textuelles utilisées

Dans ce chapitre, nous allons approfondir l'analyse des résultats obtenus avec les deux ensembles de données que nous avons utilisés dans notre étude. Bien que ces bases de données aient été introduites et expliquées en détail dans le chapitre précédent, il est essentiel de rappeler brièvement leurs caractéristiques principales pour contextualiser les résultats présentés ici. Nous nous concentrerons spécifiquement sur les informations des bases de données que nous avons utilisées dans notre étude.

Les informations détaillées de la première base de données sont résumées dans le tableau 1, et celles de la deuxième base de données sont résumées dans le tableau 2.

Caractéristique	Détail
Nom	Arabic Wikipédia corpus
Nombre de documents	Un seul document contient 9 fichiers bloc-notes
Nombre de textes	500 textes arabes
Nombre de catégories	11 catégories
Source	https://github.com/motazsaad/arWikiExtracts

Tableau 13 les informations de la première collection textuelle.

Caractéristique	Détail
Nom	Arabic Poem Classification Challenge dataset
Nombre de fichiers	Un seul fichier
Nombre de textes	25000 textes (poèmes)
Nombre de catégories	5 catégories
Taille	38.11 MB
Type	csv
Source	https://www.kaggle.com/competitions/arabic-poem-classification/data?select=poems.csv

Tableau 14 les informations de la deuxième collection textuelle.

4.5. Comparaison des algorithmes de racinisation basés sur la matrice document-terme

4.5.1. La collection textuelle « Arabic Wikipedia Corpus »

Ce tableau montre le nombre de mots après l'application de la matrice document-term sur la base de wikis.

Algorithme de racinisation	Nombre de mots
Pas de racinisation	94840
Isri	27101
Snowball	31804
Tashaphyne	39852

Tableau 15 le nombre de mots après l'application de la matrice document-term sur la base de wikis.

Les résultats montrent que sans racinisation, il y a 94 840 termes uniques, ce qui conduit à une grande dimensionnalité. En appliquant la racinisation, le nombre de mots a été réduit de manière significative.

- L'algorithme Isri a réduit le nombre de mots à 27 101, ce qui en fait l'algorithme ayant le plus réduit le nombre de mots.
- Snowball, de son côté, a réduit le nombre de mots à 31 804.
- Tashaphyne a réduit le nombre de mots à 39 852, conservant ainsi le plus grand nombre de mots parmi les trois algorithmes de racinisation.

4.5.2. La collection textuelle « Arabic Poem Classification Challenge dataset»

Ce tableau montre le nombre de mots après l'application de la matrice document-terme sur la base de poèmes.

Algorithme de racinisation	Nombre de mots
Pas de racinisation	87216
Isri	14254
Snowball	29558
Tashaphyne	15252

Tableau 16 le nombre de mots après l'application de la matrice document-terme sur la base de poèmes.

Les résultats montrent que sans racinisation, il y a 87 216 termes uniques, ce qui conduit à une grande dimensionnalité. En appliquant la racinisation, le nombre de mots a été réduit de manière significative.

- L'algorithme Isri a réduit le nombre de mots à 14 254, ce qui en fait l'algorithme ayant le plus réduit le nombre de mots.
- Snowball, de son côté, a réduit le nombre de mots à 29 558.
- Tashaphyne a réduit le nombre de mots à 15 252, conservant ainsi un nombre de mots légèrement supérieur à celui d'Isri mais bien inférieur à celui de Snowball.

4.6. Analyse des résultats des algorithmes obtenus avec les deux ensembles de données

Dans cette partie, nous analyserons les résultats des algorithmes utilisés sur nos deux ensembles de données. L'objectif est de comprendre la performance de chaque algorithme dans des contextes différents et de mettre en évidence les tendances observées.

4.6.1. Résultats de la première collection de données « Arabic Wikipédia Corpus »

Nous allons partager quelques rapports de classification obtenus en utilisant les trois racinisateurs (ISRI, SNOWBALL, TASHAPHYNE) et sans l'utilisation de racinisateurs.

Catégories	Precision	Recall	F1-score
Astronomy	0.00	0.00	0.00
Biography	0.67	0.62	0.64
Biology	1.00	0.67	0.80
Economy	0.00	0.00	0.00
Geography	0.52	1.00	0.69
History	0.67	0.36	0.47
Linguistics	0.00	0.00	0.00
Politics	0.00	0.00	0.00
Religion	0.67	0.86	0.75
Science	0.92	0.79	0.85
Technology	1.00	0.43	0.60
Accuracy			0.65
Macro avg	0.49	0.43	0.44
Weighted avg	0.64	0.65	0.61

Tableau 17 résultats de l'algorithme NB en utilisant le racinisateur ISRI.

Ce tableau montre les performances de l'algorithme de classification NB utilisant le racinisateur ISRI dans diverses catégories. Les résultats varient considérablement entre les catégories, avec des performances remarquables dans des domaines tels que la biologie, mais des scores très bas voire nuls dans des domaines comme l'astronomie et l'économie.

Catégories	Precision	Recall	F1-score
Astronomy	0.00	0.00	0.00
Biography	0.76	0.73	0.75
Biology	1.00	0.67	0.80
Economy	0.00	0.00	0.00
Geography	0.79	1.00	0.88
History	0.57	0.36	0.44
Linguistics	1.00	0.75	0.86
Politics	0.00	0.00	0.00
Religion	0.75	0.86	0.80
Science	0.81	0.93	0.87
Technology	0.78	1.00	0.88
Accuracy			0.77
Macro avg	0.59	0.57	0.57
Weighted avg	0.73	0.77	0.74

Tableau 18 résultats de l'algorithme MLP en utilisant le racinisateur ISRI.

Ce tableau montre les résultats de l'algorithme de classification MLP utilisant le racinisateur ISRI dans diverses catégories. Les performances varient selon la catégorie, avec des scores élevés dans des domaines tels que la biologie, la linguistique et la technologie, mais des performances faibles voire nulles dans des domaines comme l'astronomie, l'économie et la politique.

Catégories	Precision	Recall	F1-score
Astronomy	0.00	0.00	0.00
Biography	0.65	0.58	0.61
Biology	0.00	0.00	0.00
Economy	0.00	0.00	0.00
Geography	0.32	1.00	0.48
History	0.00	0.00	0.00
Linguistics	0.00	0.00	0.00
Politics	0.00	0.00	0.00
Religion	0.50	0.14	0.22
Science	0.00	0.00	0.00
Technology	1.00	0.29	0.44
Accuracy			0.41
Macro avg	0.22	0.18	0.16
Weighted avg	0.35	0.41	0.32

Tableau 19 résultats de l'algorithme SVM en utilisant le racinisateur SNOWBALL.

Ce tableau présente les résultats de l'algorithme SVM utilisant le racinisateur SNOWBALL dans différentes catégories. Les performances varient considérablement, avec des scores faibles voire nuls dans la plupart des catégories, à l'exception notable de la technologie, où l'algorithme a obtenu une précision de 100 %. Cependant, dans l'ensemble, les performances globales de l'algorithme sont assez basses.

Catégories	Precision	Recall	F1-score
Astronomy	0.50	0.50	0.50
Biography	0.55	0.81	0.66
Biology	1.00	0.33	0.50
Economy	0.00	0.00	0.00
Geography	0.87	0.87	0.87
History	0.60	0.27	0.37
Linguistics	0.67	0.50	0.57
Politics	0.00	0.00	0.00
Religion	0.83	0.71	0.77
Science	0.79	0.79	0.79
Technology	0.86	0.86	0.86
Accuracy			0.70
Macro avg	0.61	0.51	0.53
Weighted avg	0.70	0.70	0.68

Tableau 20 résultats de l'algorithme BOOSTING en utilisant le racinisateur SNOWBALL.

Ce tableau présente les performances de l'algorithme boosting utilisant le racinisateur SNOWBALL dans différentes catégories. Les résultats montrent une variabilité significative entre les catégories, avec des scores élevés dans des domaines tels que la géographie, la religion et la technologie, mais des performances plus modestes dans d'autres domaines comme l'histoire et la biologie. Globalement, l'algorithme semble avoir une précision et un rappel raisonnables, bien que des améliorations pourraient être nécessaires dans certaines catégories pour augmenter le score F1 et l'exactitude globale.

Catégories	Precision	Recall	F1-score
Astronomy	0.00	0.00	0.00
Biography	0.70	0.62	0.65
Biology	1.00	0.67	0.80
Economy	0.00	0.00	0.00
Geography	0.56	1.00	0.72
History	0.67	0.36	0.47
Linguistics	0.00	0.00	0.00
Politics	0.00	0.00	0.00
Religion	0.67	0.86	0.75
Science	0.92	0.86	0.89
Technology	0.83	0.71	0.77
Accuracy			0.68
Macro avg	0.49	0.46	0.46
Weighted avg	0.65	0.68	0.64

Tableau 21 résultats de l'algorithme NB en utilisant le racinisateur TASHAPHYNE.

Ce tableau présente les performances de l'algorithme de classification NB utilisant le racinisateur TASHAPHYNE dans différentes catégories. Les résultats varient considérablement selon la catégorie. Le modèle obtient d'excellents scores en biologie et en science, mais des scores beaucoup plus faibles dans des domaines tels que l'astronomie et la linguistique.

Catégories	Precision	Recall	F1-score
Astronomy	1.00	0.50	0.67
Biography	0.57	0.81	0.67
Biology	1.00	0.33	0.50
Economy	0.00	0.00	0.00
Geography	0.79	1.00	0.88
History	0.57	0.36	0.44
Linguistics	0.00	0.00	0.00
Politics	0.00	0.00	0.00
Religion	0.62	0.71	0.67
Science	1.00	0.71	0.83
Technology	0.80	0.57	0.67
Accuracy			0.69
Macro avg	0.58	0.45	0.48
Weighted avg	0.68	0.69	0.66

Tableau 22 résultats de l'algorithme BOOSTING en utilisant le racinisateur TASHAPHYNE.

Ce tableau présente les performances de l'algorithme boosting utilisant le racinisateur TASHAPHYNE dans différentes catégories. Les résultats montrent une variabilité significative entre les catégories, avec des scores élevés dans des domaines tels que la géographie et la science, mais des performances plus modestes dans d'autres domaines comme l'histoire et la biologie.

Catégories	Precision	Recall	F1-score
Astronomy	0.00	0.00	0.00
Biography	0.82	0.35	0.49
Biology	0.00	0.00	0.00
Economy	0.00	0.00	0.00
Geography	0.26	1.00	0.42
History	0.00	0.00	0.00
Linguistics	0.00	0.00	0.00
Politics	0.00	0.00	0.00
Religion	0.50	0.14	0.22
Science	0.00	0.00	0.00
Technology	0.00	0.00	0.00
Accuracy			0.33
Macro avg	0.14	0.14	0.10
Weighted avg	0.31	0.33	0.24

Tableau 23 résultats de l'algorithme SVM sans l'utilisation de racinisateur.

Ce tableau présente les performances de l'algorithme SVM sans l'utilisation de racinisateur dans différentes catégories. Les résultats montrent une variabilité significative, avec des scores très bas voire nuls dans la plupart des catégories, à l'exception notable de la géographie où le rappel est de 100 %. Globalement, l'exactitude pondérée suggère une performance très basse du modèle.

Catégories	Precision	Recall	F1-score
Astronomy	0.00	0.00	0.00
Biography	0.79	0.88	0.84
Biology	1.00	0.67	0.80
Economy	0.00	0.00	0.00
Geography	0.76	0.96	0.85
History	0.80	0.36	0.50
Linguistics	1.00	0.75	0.86
Politics	0.00	0.00	0.00
Religion	0.75	0.86	0.80
Science	0.76	0.93	0.84
Technology	1.00	0.86	0.92
Accuracy			0.79
Macro avg	0.62	0.57	0.58
Weighted avg	0.77	0.79	0.76

Tableau 24 résultats de l'algorithme MLP sans l'utilisation de racinisateur.

Ce tableau présente les performances de l'algorithme MLP sans l'utilisation de racinisateur dans différentes catégories. Les résultats montrent une variabilité significative des scores, avec des performances élevées dans des domaines comme la biologie, la géographie, la linguistique, la religion et la technologie, mais des performances plus modestes dans d'autres domaines comme l'astronomie, l'économie et la politique. Globalement, l'exactitude pondérée suggère une performance raisonnable du modèle.

4.6.2. Résultats de la deuxième collection de données « Arabic Poem Classification Challenge dataset»

Nous allons partager quelques rapports de classification obtenus en utilisant les trois racinisateurs (ISRI, SNOWBALL, TASHAPHYNE) et sans l'utilisation de racinisateurs.

Catégories	Precision	Recall	F1-score
العصر الأندلسي	0.43	0.49	0.46
العصر حديث	0.44	0.61	0.51
العصر الايوبي	0.48	0.31	0.38
العصر العباسي	0.45	0.36	0.40
العصر المملوكي	0.60	0.62	0.61
Accuracy			0.48
Macro avg	0.48	0.48	0.47
Weighted avg	0.48	0.48	0.48

Tableau 25 résultats de l'algorithme NB en utilisant le racinisateur ISRI.

Ce tableau présente les performances de l'algorithme de classification NB utilisant le racinisateur ISRI pour différentes périodes historiques en arabe. Les résultats montrent des variations entre les catégories, avec des scores F1 allant de 0.38 pour "العصر الايوبي" à 0.61 pour "العصر المملوكي". La précision, le rappel et le F1-score moyens indiquent une performance modérée de l'algorithme.

Catégories	Precision	Recall	F1-score
العصر الأندلسي	0.40	0.33	0.37
العصر حديث	0.47	0.48	0.47
العصر الايوبي	0.37	0.37	0.37
العصر العباسي	0.40	0.57	0.47
العصر المملوكي	0.69	0.52	0.59
Accuracy			0.46
Macro avg	0.47	0.45	0.45
Weighted avg	0.47	0.46	0.46

Tableau 26 résultats de l'algorithme MLP en utilisant le racinisateur ISRI.

Ce tableau présente les performances de l'algorithme MLP utilisant le racinisateur ISRI pour différentes périodes historiques en arabe. Les résultats montrent des variations significatives entre les catégories, avec des scores F1 allant de 0.37 pour "العصر الأندلسي" et "العصر الايوبي" à 0.59 pour "العصر المملوكي". La précision, le rappel et le F1-score moyens indiquent une performance modérée de l'algorithme.

Catégories	Precision	Recall	F1-score
العصر الأندلسي	0.54	0.07	0.12
العصر حديث	0.47	0.22	0.30
العصر الايوبي	0.48	0.26	0.34
العصر العباسي	0.28	0.83	0.42
العصر المملوكي	0.56	0.47	0.51
Accuracy			0.38
Macro avg	0.47	0.37	0.34
Weighted avg	0.47	0.38	0.34

Tableau 27 résultats de l'algorithme SVM en utilisant le racinisateur SNOWBALL.

Ce tableau présente les performances de l'algorithme SVM utilisant le racinisateur SNOWBALL pour différentes périodes historiques en arabe. Les résultats montrent des variations significatives entre les catégories, avec des scores F1 allant de 0.12 pour " العصر الأندلسي" à 0.51 pour "العصر المملوكي". La catégorie "العصر العباسي" a un rappel élevé de 0.83 mais une précision plus faible, ce qui affecte son score F1.

Catégories	Precision	Recall	F1-score
العصر الأندلسي	0.47	0.23	0.30
العصر حديث	0.37	0.33	0.35
العصر الايوبي	0.36	0.37	0.37
العصر العباسي	0.33	0.64	0.44
العصر المملوكي	0.68	0.47	0.56
Accuracy			0.41
Macro avg	0.44	0.41	0.40
Weighted avg	0.45	0.41	0.41

Tableau 28 résultats de l'algorithme BOOSTING en utilisant le racinisateur SNOWBALL.

Ce tableau présente les performances de l'algorithme boosting utilisant le racinisateur SNOWBALL pour différentes périodes historiques en arabe. Les résultats montrent des variations significatives entre les catégories, avec des scores F1 allant de 0.30 pour " العصر الأندلسي" à 0.56 pour "العصر المملوكي". La catégorie "العصر العباسي" a un rappel élevé de 0.64 mais une précision plus faible, ce qui affecte son score F1.

Catégories	Precision	Recall	F1-score
العصر الأندلسي	0.40	0.52	0.45
العصر حديث	0.42	0.59	0.49
العصر الايوبي	0.56	0.25	0.35
العصر العباسي	0.41	0.29	0.34
العصر المملوكي	0.61	0.68	0.64
Accuracy			0.47
Macro avg	0.48	0.47	0.46
Weighted avg	0.48	0.47	0.46

Tableau 29 résultats de l'algorithme NB en utilisant le racinisateur TASHAPHYNE.

Ce tableau présente les performances de l'algorithme de classification NB utilisant le racinisateur TASHAPHYNE pour différentes périodes historiques en arabe. Les résultats montrent des variations notables entre les catégories, avec des scores F1 allant de 0.34 pour "العصر العباسي" à 0.64 pour "العصر المملوكي". Les catégories "العصر الأندلسي" et "العصر حديث" montrent des performances intermédiaires, avec des scores F1 de 0.45 et 0.49 respectivement.

Catégories	Precision	Recall	F1-score
العصر الأندلسي	0.40	0.18	0.24
العصر حديث	0.37	0.24	0.29
العصر الايوبي	0.36	0.30	0.33
العصر العباسي	0.28	0.69	0.40
العصر المملوكي	0.70	0.44	0.54
Accuracy			0.38
Macro avg	0.42	0.37	0.36
Weighted avg	0.43	0.38	0.37

Tableau 30 résultats de l'algorithme BOOSTING en utilisant le racinisateur TASHAPHYNE.

Ce tableau présente les performances de l'algorithme boosting utilisant le racinisateur TASHAPHYNE pour différentes périodes historiques en arabe. Les résultats montrent des variations significatives entre les catégories, avec des scores F1 allant de 0.24 pour " العصر الأندلسي" à 0.54 pour "العصر المملوكي". La catégorie "العصر العباسي" a un rappel élevé de 0.69 mais une précision plus faible, ce qui affecte son score F1.

Catégories	Precision	Recall	F1-score
العصر الأندلسي	0.43	0.06	0.10
العصر حديث	0.48	0.16	0.24
العصر الايوبي	0.45	0.23	0.30
العصر العباسي	0.28	0.85	0.42
العصر المملوكي	0.49	0.41	0.45
Accuracy			0.35
Macro avg	0.43	0.34	0.30
Weighted avg	0.43	0.35	0.31

Tableau 31 résultats de l'algorithme SVM sans l'utilisation de racinisateur.

Ce tableau présente les performances de l'algorithme SVM sans l'utilisation de racinisateur pour différentes périodes historiques en arabe. Les résultats montrent des variations importantes entre les catégories, avec des scores F1 allant de 0.10 pour "العصر الأندلسي" à 0.45 pour " العصر المملوكي". La catégorie "العصر العباسي" se distingue par un rappel élevé de 0.85, mais avec une précision plus faible, ce qui lui donne un score F1 de 0.42.

Catégories	Precision	Recall	F1-score
العصر الأندلسي	0.46	0.37	0.41
العصر حديث	0.50	0.45	0.48
العصر الايوبي	0.42	0.48	0.45
العصر العباسي	0.36	0.53	0.43
العصر المملوكي	0.68	0.47	0.55
Accuracy			0.46
Macro avg	0.48	0.46	0.46
Weighted avg	0.49	0.46	0.47

Tableau 32 résultats de l'algorithme MLP sans l'utilisation de racinisateur.

Ce tableau présente les performances de l'algorithme MLP sans l'utilisation de racinisateur pour différentes périodes historiques en arabe. Les résultats montrent des variations notables entre les catégories, avec des scores F1 allant de 0.41 pour "العصر الأندلسي" à 0.55 pour "العصر المملوكي".

Les scores intermédiaires incluent 0.48 pour "العصر حديث", 0.45 pour "العصر الايوبي" et 0.43 pour "العصر العباسي".

4.7. Comparaison graphique des résultats : collection textuelle 1 et 2

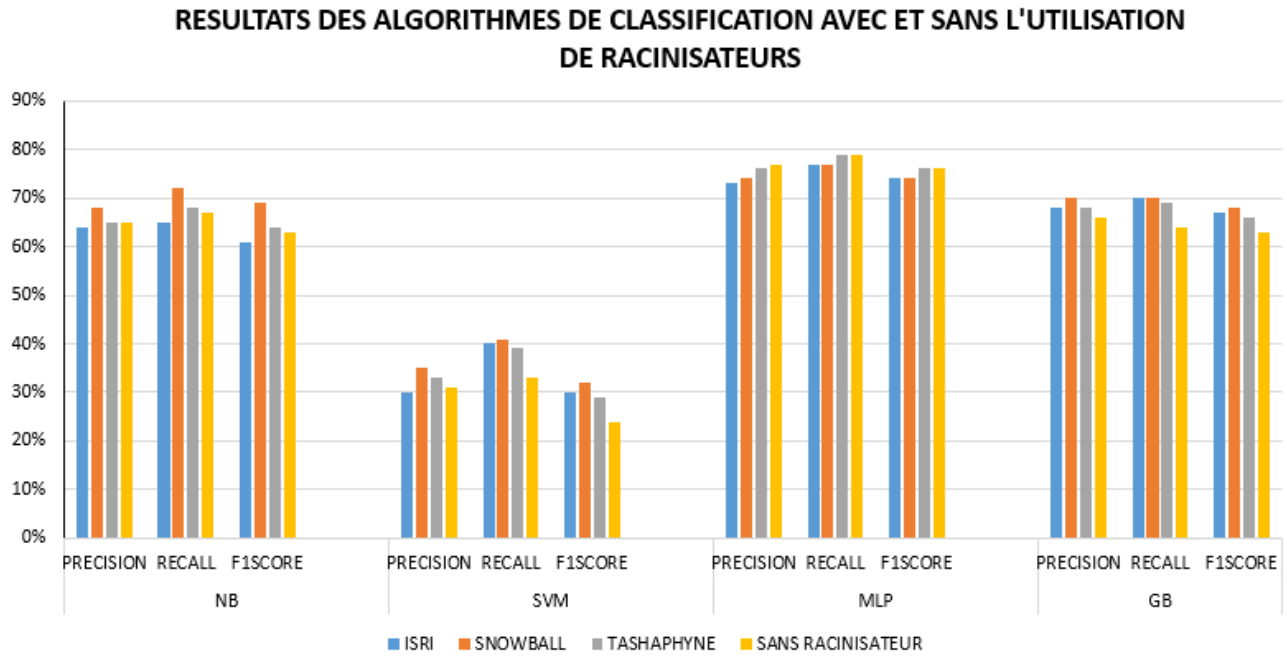

Figure 14 présentations graphiques des résultats de la première base de données « Arabic Wikipedia Corpus ».

Le graphique en barres montre les résultats des algorithmes de classification avec et sans l'utilisation de racinisateurs. Les algorithmes de classification évalués sont Naive Bayes (NB), Support Vector Machine (SVM), Multi-Layer Perceptron (MLP) et Gradient Boosting (GB). Les métriques de performance affichées sont la précision, le rappel et le score F1. Chaque algorithme est testé avec différents racinisateurs : ISRI, Snowball, Tashaphyne, et sans racinisateur (Sans Racinisateur). Ces résultats sont obtenus sur la collection textuelle Arabic Wikipedia Corpus.

Le Multi-Layer Perceptron (MLP) se distingue comme le meilleur algorithme de classification, présentant les meilleures performances globales avec des valeurs de précision, de rappel et de score F1 très élevées (environ 80%) pour tous les racinisateurs ainsi que sans racinisateur. Cet algorithme démontre ainsi une efficacité supérieure dans ce contexte. Quant aux racinisateurs, Snowball se distingue légèrement des autres, en particulier pour les algorithmes Naive Bayes (NB) et Multi-Layer Perceptron (MLP), en améliorant légèrement la précision et le rappel. Cependant, il est important de noter que l'absence de racinisateur ne nuit pas aux performances dans plusieurs cas, notamment pour les algorithmes MLP et Gradient Boosting (GB). Cela suggère que l'utilisation d'un racinisateur n'est pas toujours nécessaire pour obtenir de bons résultats.

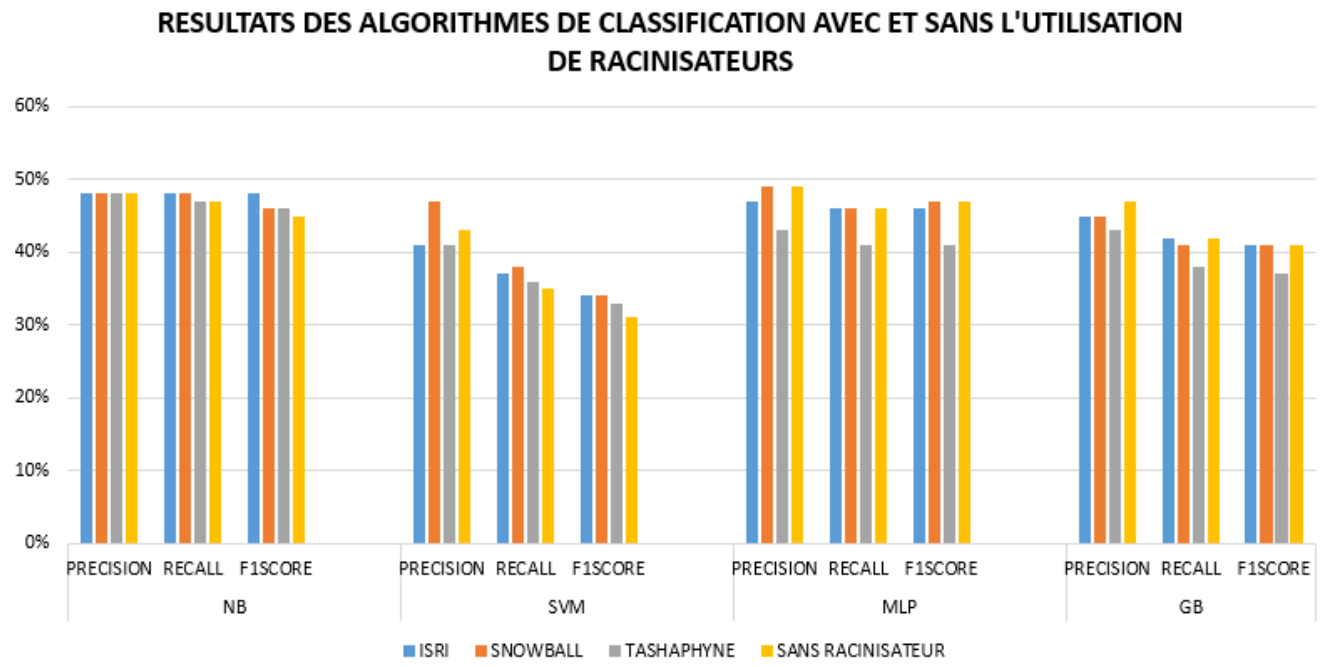

Figure 15 présentations graphiques des résultats de la deuxième base de données « Arabic Poem Classification Challenge dataset».

Le graphique en barres montre les résultats des algorithmes de classification avec et sans l'utilisation de racinisateurs. Les algorithmes de classification évalués sont Naive Bayes (NB), Support Vector Machine (SVM), Multi-Layer Perceptron (MLP) et Gradient Boosting (GB). Les métriques de performance affichées sont la précision, le rappel et le score F1. Chaque algorithme est testé avec différents racinisateurs : ISRI, Snowball, Tashaphyne, et sans racinisateur (Sans Racinisateur). Ces résultats sont obtenus sur la collection textuelle Arabic Poem Classification Challenge dataset.

D'après les résultats présentés, le meilleur algorithme et racinisateur est NB avec le racinisateur ISRI**.** Cet algorithme obtient les meilleurs résultats en termes de précision et de F1-score, tant avec qu'sans racinisateur. Il est également important de noter que l'utilisation d'un racinisateur n'améliore pas toujours les performances des algorithmes de classification. Dans ce cas, l'utilisation du racinisateur ISRI avec l'algorithme NB a permis d'améliorer significativement les performances de l'algorithme.

Il est crucial de noter que le type de collection textuelle utilisé influence significativement les résultats des algorithmes de classification. Cependant, lors de l'utilisation d'une autre collection textuelle, telle que des poèmes, les racinisateurs peuvent ne pas fonctionner de manière optimale. Cela est dû à la nature particulière du langage poétique, qui diffère considérablement du texte encyclopédique. Les racinisateurs, conçus principalement pour des textes structurés et informatifs comme ceux de Wikipedia, peuvent perdre en efficacité lorsqu'ils sont appliqués à des textes poétiques, où les structures et les expressions sont plus variées et moins standardisées. C'est pourquoi les résultats peuvent varier et apparaître moins performants lorsque des racinisateurs sont appliqués à des collections de poèmes.

4.8. Conclusion

En conclusion, notre étude sur la classification de textes a révélé des insights significatifs sur l'efficacité des algorithmes de machine learning et de deep learning en fonction de l'application de racinisateurs et de la nature des ensembles de données utilisés. Nous avons testé quatre algorithmes de classification sur des textes provenant de deux collections distinctes, notamment l'Arabic Wikipedia Corpus et une collection de poèmes. Les résultats montrent que le type de collection textuelle influence fortement les performances des algorithmes, en particulier concernant l'utilisation de racinisateurs.

Cependant, il est important de noter que les racinisateurs, conçus pour des textes structurés et informatifs, tels que ceux de Wikipedia, ont montré des limitations lorsqu'ils sont appliqués à des collections de poèmes. Cela souligne l'importance de considérer la nature du texte lors de l'application de méthodes de prétraitement. Nos résultats suggèrent que l'adaptation des techniques de classification et de prétraitement aux caractéristiques spécifiques des données textuelles peut améliorer significativement les performances des modèles.

Enfin, cette étude ouvre des perspectives d'amélioration futures pour la classification de textes, notamment en explorant des techniques de prétraitement plus adaptées aux différentes natures de textes et en continuant à affiner les algorithmes pour mieux capturer les nuances linguistiques spécifiques.

Conclusion générale

Conclusion générale

En résumé, ce mémoire met en évidence l'importance cruciale de la classification automatique des textes arabes face à l'explosion des données textuelles. La langue arabe, avec son riche patrimoine historique et culturel, exige des approches sophistiquées pour organiser et exploiter efficacement les informations provenant de sources variées, des manuscrits anciens aux publications numériques contemporaines.

Notre étude a détaillé les étapes essentielles de la classification des textes, en commençant par le prétraitement des données. Cette phase est indispensable pour transformer des informations non structurées en données prêtes à l'analyse. Nous avons également examiné les applications pratiques de la classification automatique dans divers domaines, démontrant ainsi son utilité pour la recherche académique, la gestion de l'information et la veille médiatique. Les défis spécifiques à la langue arabe, notamment sa complexité morphologique et sa richesse lexicale, ont été analysés, soulignant les obstacles à surmonter pour atteindre une classification efficace.

Le chapitre consacré à la racinisation a permis d'explorer en profondeur les techniques de stemming, essentielles pour le traitement des textes arabes. Nous avons étudié les différentes méthodes de stemming, en mettant l'accent sur leurs applications spécifiques à la langue arabe, ce qui est crucial pour améliorer la précision de la classification.

La méthodologie de notre recherche, de la collecte des données à l'analyse des résultats, a été soigneusement structurée pour assurer une base solide et rigoureuse. Les sources de données variées et les processus rigoureux de collecte garantissent la fiabilité des analyses. L'importance du prétraitement des données et du choix des algorithmes a été mise en lumière, montrant leur impact significatif sur les performances globales de la classification.

Les résultats obtenus démontrent que la combinaison de techniques avancées de prétraitement, de méthodes de stemming adaptées et d'algorithmes performants peut considérablement améliorer l'organisation et la recherche de textes en langue arabe. Les outils et méthodes utilisés ont été évalués de manière rigoureuse, offrant des insights précieux pour l'amélioration continue de ces techniques.

Ce mémoire apporte une contribution significative au domaine de la classification automatique des textes arabes. En relevant les défis spécifiques posés par la langue arabe, nous avons développé et testé des approches novatrices qui facilitent l'accès à l'information et la découverte de connaissances dans ce riche corpus textuel. Notre travail se distingue par son analyse approfondie et ses contributions originales, ouvrant de nouvelles perspectives pour la préservation, l'organisation et la diffusion de la littérature arabe. Nous espérons que cette recherche servira de référence pour les futurs travaux, encourageant l'innovation et l'amélioration continue des méthodes de classification des textes en langue arabe.

Bibliographie

1. Mohammed, Mr. Aouine. ***CATEGORISATION AUTOMATIQUE DE TEXTE ARABE.*** **GUELMA : s.n., 2009.**

2. BELAZZOUG, Mouhoub. Apprentissage statistique pour l'extraction des relations à partir de textes. Sétif : s.n., 2021.

3. **Chapitre V: Arbre de décision. Introduction à l'apprentissage automatique. [En ligne] https://projeduc.github.io/intro_apprentissage_automatique/arbres.html.**

4. Qu'est ce que l'algorithme KNN ? [En ligne] https://datascientest.com/knn.

5. Naive Bayes Classifier pour Machine Learning. [En ligne] https://mrmint.fr/naive-bayes-classifier.

6. SVM. [En ligne] https://dataanalyticspost.com/Lexique/svm/.

7. Prévision de Défaillance Des entreprises ***: Apport des Réseaux de Neurones Artificiels.*** **MESK, Siham LOTFI and Hicham. 2021.**

8. Que sont les réseaux neuronaux ? [En ligne] https://www.ibm.com/fr-fr/topics/neural-networks.

9. Qu'est-ce que le boosting ? [En ligne] https://aws.amazon.com/fr/what-is/boosting/.

10. Présentation (commande MLP). [En ligne] https://www.ibm.com/docs/fr/spss-statistics/saas?topic=mlp-overview-command.

11. The Porter stemming algorithm. [En ligne] https://snowballstem.org/algorithms/porter/stemmer.html.

12. BOUKHARI, KABIL. ***Un Nouvel Algorithme de Stemmatisation pour l'Indexation Automatique de Documents non-structurés : Stemmer SAID*** **. MONASTIR : s.n., 2013.**

13. ***ARABIC WORDS STEMMING APPROACH USING ARABIC WORDNET.*** **Abdel Hamid Kreaa, Ahmad S Ahmad and Kassem Kabalan. Latakia, Syria : s.n., 2014.**

14. ***A Comparative Survey on Arabic Stemming: Approaches and Challenges.*** **Mustafa, Mohammad. Tabuk, SA : s.n., 2017.**

15. ***Arabic light-based stemmer using new rules.*** **Alshalabi, Hamood. Bangi, Malaysia : s.n., 2022.**

16. ***Conditional Arabic Light Stemmer: CondLight.*** **Yaser Al-Lahham, Khawlah Matarneh, and Mohammad Hassan. Zarqa University, Jordan : s.n., 2018.**

17. ***WikiDocsAligner: an off-the-shelf Wikipedia Documents Alignment Tool.*** **Saad, Motaz. Gaza, Palestine : s.n., 2017.**

18. Arabic Poem Classification. [En ligne] https://www.kaggle.com/competitions/arabic-poem-classification/data.

19. Python : qu'est-ce que c'est ? [En ligne] https://www.futura-sciences.com/tech/definitions/informatique-python-19349/.

20. Anaconda pour Python - Présentation et installation. [En ligne] https://www.jedha.co/formation-python/ananconda-python.

21. Jupyter Notebook. [En ligne] https://www.databricks.com/glossary/jupyter-notebook.

22. Natural Language Toolkit. [En ligne] https://www.nltk.org/.

23. Scikit Learn: Discover the Python library dedicated to Machine Learning. [En ligne] https://datascientest.com/en/scikit-learn-discover-the-python-library-dedicated-to-machine-learning.

24. Pandas : la bibliothèque Python dédiée à la Data Science. [En ligne] https://datascientest.com/pandas-python-data-science.

25. Le module OS Python. [En ligne] https://www.tresfacile.net/le-module-os-python/.