République algérienne démocratique et populaire
Ministre de l'enseignement supérieur et de la recherche scientifique
Université Saad Dahlab blida 1
Faculté Des Sciences
Département d'Informatique

Mémoire

Présenté pour l'obtention du diplôme de Master 2
En : Informatique
Spécialité : Ingénierie des Logiciels

Réalisé par
Mahi Yacine
Khelif Sami

THÈME

Proposition d'une approche basée sur l'apprentissage supervisée pour la classification multi-étiquettes du texte arabe

---

Soutenu le : devant le jury compposé de :

M.	CHIKHI Nacim	Présidente
M.	FERFERA Soufiane	Examinateur
Mme.	CHERIGUENE Soraya	Promotrice

2022/2023

Remerciements

Ce travail de mémoire met fin à notre cycle de Master et n'a pu aboutir qu'avec l'aide et les encouragements de plusieurs personnes. Nous espérons pouvoir exprimer notre profonde gratitude envers toutes ces personnes. D'abord, à Allah tout puissant, qui nous a aidé, comblé d'amour et nous a donné la force de continuer aux moments les plus durs de toute notre vie.

Nos sincères remerciements vont à nos enseignants et membres de l'équipe pédagogique du département d'informatique qui ont contribué, directement ou indirectement de près ou de loin à la réalisation de ce travail. Leurs leçons, commentaires, suggestions et conseils ont influencé notre parcours académique.

Avec un sincère sentiment de respect, nous remercions notre promotrice, Mme. CHERIGUENE Soraya, nous sommes reconnaissants pour ses efforts dans notre travail et ses qualités pédagogiques tout au long de cette période difficile, et qui nous serviront aussi à l'avenir.

Nous tenons à remercier les membres du jury pour leur aimable participation à l'évaluation de ce travail, nous en sommes honorés et nous leur exprimons notre profonde reconnaissance.

Nos vifs remerciements de reconnaissance sont également adressés à nos chers parents qui ont sacrifié énormément pour nous. Ils ont toujours été là pour nous encourager et nous aider avec grand amour. Il va de soi pour nos grands-parents qui nous ont comblé de joie durant notre enfance. Une pensée à nos grands parents « Chikhaoui Djilalli, Benkortbi Abderezak, Mahi Ahmed et Moumedji Rabea », ils auraient été fier de nous voir en arriver là. Nous tenons aussi à présenter notre gratitude et remercîments envers les familles « Khelif et Mahi », y compris les cousins, cousines, frères et sœurs de nous avoir aider moralement avec leurs paroles, conseils et expériences.

Un grand remerciement va aussi à Omar et Islem, nos camarades avec qui nous avons pu travailler plusieurs fois durant cette période. En fin que les amis, frères et sœurs dont les noms ne sont pas cités ne nous tiennent pas rigueur, nos pensées vont aussi vers eux.

Dédicace

Je dédis ce travail

A la personne la plus chère à mon cœur, ma mère qui m'a encouragé à poursuivre mes rêves et soutenu dans chaque étape de ma vie. Que Dieu le tout puissant veille sur elle et la protège.

A mon superbe père qui m'indique toujours la bonne voie, qui a fait de moi ce que je suis aujourd'hui.

A mon cher frère Mohamed et mes cheres soeurs Nacera et Yasmina pour leurs appui et leur encouragement.

A mes chers amis.

A ma grand-mère et à toute ma famille maternelle et paternelle.

A mon chère et fidèle binôme et ami Sami Merci de m'avoir encouragé et surtout d'avoir été toujours là pour moi dans les moments les plus durs, ainsi qu'à toute sa famille.

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible,

Merci d'être toujours là pour moi.

Yacine.

Je dédis ce travail

A la personne la plus chère à mon cœur, ma mère qui m'a encouragé à poursuivre mes rêves et soutenu dans chaque étape de ma vie. Que Dieu le tout puissant veille sur elle et la protège.

A mon père qui me montre toujours le droit chemin, qui a fait de moi ce que je suis aujourd'hui, et qui fait tout pour me rendre heureux.

A mon cher frère Wassim et ma soeur Sabrina pour leurs appuis et leurs encouragements.

A mes chers amis.

A mes grands-parents et à toute ma famille maternelle et paternelle.

A mon chère et fidèle binôme et ami Yacine Merci de m'avoir soutenu, ainsi qu'à toute sa famille.

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible,

Merci d'être toujours là pour moi.

Sami.

Résumé

Ce mémoire de fin d'étude se concentre sur la classification multi-étiquettes en arabe dans le domaine du traitement automatique du langage naturel (TALN). Nous proposons une approche avant-gardiste qui exploite la corrélation entre les différentes classes du jeu de données pour améliorer les performances de classification. Malgré les défis rencontrés, notre travail contribue à l'avancement du TALN en arabe en offrant de nouvelles perspectives pour la recherche future. Les résultats de nos expérimentations soulignent les complexités spécifiques de la langue arabe et mettent en évidence les obstacles inhérents à la classification du texte et à l'implémentation technique. Notre mémoire constitue une référence précieuse pour les chercheurs et praticiens intéressés par la classification multi-étiquettes en arabe, en fournissant des enseignements sur les difficultés rencontrées et les possibilités d'amélioration.

Mots clés : classification multi-étiquettes, traitement automatique du langage naturel de la langue arabe, corrélation, classification du texte.

Abstract

This final study thesis focuses on multi-label classification in Arabic within the field of Natural Language Processing (NLP). We propose an innovative approach that leverages the correlation between different classes in the dataset to enhance classification performance. Despite the challenges encountered, our work contributes to the advancement of Arabic NLP by providing new perspectives for future research. The results of our experiments highlight the specific complexities of the Arabic language and shed light on the inherent obstacles in text categorization and technical implementation. Our thesis serves as a valuable reference for researchers and practitioners interested in multi-label classification in Arabic, offering insights into the encountered difficulties and possibilities for improvement.

Keywords : multi-label classification, natural language processing of the Arabic language, correlation, text classification.

الملخص

هذا المذكرة الختامية تركز على التصنيف متعدد العلامات في اللغة العربية في مجال معالجة اللغة الطبيعية. نقدم نهجاً مبتكراً يستغل الترابط بين الفئات المختلفة في مجموعة البيانات لتحسين أداء التصنيف. على الرغم من التحديات التي واجهناها، يسهم عملنا في تقدم معالجة اللغة الطبيعية في اللغة العربية عن طريق توفير آفاق جديدة للبحث المستقبلي. تسلط نتائج تجاربنا الضوء على تعقيدات اللغة العربية الخاصة وتسلط الضوء على العقبات الجوهرية في تصنيف النص وتنفيذ التقنية. تمثل مذكرتنا مرجعًا قيمًا للباحثين والممارسين المهتمين بالتصنيف متعدد العلامات باللغة العربية، حيث توفر نظرة على التحديات التي واجهناها وإمكانيات التحسين.

الكلمات المفتاحية : تصنيف متعدد العلامات، معالجة اللغة الطبيعية للغة العربية، الترابط، تصنيف النصوص

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Dans un monde en constante évolution, le traitement automatique du langage naturel (TALN) occupe une place prépondérante. Les avancées technologiques et l'omniprésence des données textuelles ont engendré de nombreux défis, notamment dans le domaine de la classification multi-label. Cette forme de classification permet d'attribuer à chaque instance plusieurs étiquettes parmi un ensemble de classes prédéfinies, offrant ainsi une approche plus réaliste pour représenter la complexité du langage.

Le domaine spécifique que nous abordons dans ce mémoire de fin d'étude concerne la classification multi-label appliquée à la langue arabe. Cette langue, riche et complexe, présente des particularités qui nécessitent des approches spécifiques pour obtenir des résultats précis et significatifs. Notre objectif principal est de proposer une nouvelle approche qui exploite la corrélation existante entre les différentes classes du jeu de données, afin d'améliorer les performances de classification multi-label en arabe.

L'exploitation de la corrélation entre les classes du jeu de données constitue une piste prometteuse pour améliorer les résultats de classification multi-label. En effet, les données textuelles en langue arabe comportent souvent des relations sémantiques et syntaxiques complexes entre les différentes étiquettes. En considérant ces relations, notre approche vise à mieux capturer la richesse et la diversité des textes en arabe, et ainsi améliorer la précision et la cohérence de la classification multi-label.

PLAN DU MÉMOIRE

Ce mémoire de fin d'étude se structure de la manière suivante :

- **Chapitre 1 :** Catégorisation automatique du texte.
- **Chapitre 2 :** Conception.
- **Chapitre 3 :** Implémentations et résultats.

Le présent mémoire parviendra à son achèvement avec l'intégration d'une conclusion générale, qui récapitulera les points importants abordés tout en exposant des perspectives d'amélioration de notre travail.

Ce mémoire vise à contribuer à l'amélioration de la classification multi-label en langue arabe en proposant une approche novatrice basée sur l'exploitation de la corrélation entre les différentes classes du jeu de données. Nous espérons ainsi ouvrir de nouvelles perspectives pour le traitement automatique du langage arabe et apporter des avancées significatives dans ce domaine en plein essor.

Chapitre 1

Classification automatique du texte

1.1 Introduction

La classification automatique du texte est une tâche fondamentale du Traitement Automatique du de la langue (TAL) qui consiste à attribuer automatiquement des étiquettes prédéfinies à des documents textuels en fonction de leurs contenus. La classification classique à une seule étiquette assigne à chaque document textuel une seule classe prédéfinie, tandis que les objets du monde réel peuvent revêtir plusieurs significations sémantiques simultanément. Afin de surmonter ce défi, une approche courante est d'adopter la classification multi-étiquettes, où plusieurs étiquettes peuvent être associées à une même instance [1].

Le présent chapitre est divisé en deux parties principales. Dans la première partie, nous commencerons par examiner les différentes techniques utilisées pour la classification du texte ainsi que les adaptations spécifiques nécessaires pour la classification du texte arabe. Cette dernière présente des défis uniques qui nécessitent des méthodes et des approches adaptées. Nous nous intéresserons dans la deuxième partie à un aspect spécifique de la classification du texte, à savoir la classification multi-étiquettes. Nous explorerons les différentes méthodes proposées par ce type de classification en exposant quelques travaux sur la classification multi-étiquettes de texte arabe [1].

1.2 Classification automatique du texte

La classification automatique de documents est une problématique bien connue en informatique, qui consiste à assigner un document à plusieurs classes spécifiques. La nature des documents en question influe sur la manière dont ce problème est abordé, la classification de textes diffère de celle des documents images, vidéos ou encore audio. La classification du texte est généralement effectuée à l'aide de techniques d'apprentissage automatique, où un modèle est formé sur un ensemble de données étiquetées pour pouvoir généraliser et classer de nouveaux textes non étiquetés [1].

L'objectif principal de la classification automatique est de permettre une analyse, une organisation et une compréhension efficaces d'un texte brut, ce qui conduit à une meilleure structuration et exploitation des données textuelles massives. Cette tâche joue un rôle clé dans de nombreux domaines tels que la détection de spam, l'organisation des documents, la classification des documents multimédia, l'analyse des sentiments et la fouille d'opinion [1].

1.3 Processus de la classification du texte

Le processus de classification implique la construction d'un système de prédiction qui prend un texte en entrée et lui attribue une ou plusieurs étiquettes en sortie. Pour déterminer la classe à laquelle un texte est associé, un ensemble d'étapes sont généralement suivies. Ces étapes sont cruciales pour transformer les données textuelles brutes en informations significatives. Le processus de classification de texte comprend plusieurs phases, comme le prétraitement, la représentation du texte, le choix de l'algorithme d'apprentissage, ainsi que les évaluations des résultats obtenus pour garantir une bonne généralisation du modèle appris.

Dans l'étapes du prétraitement le texte est nettoyé de tout caractére indésirable afin d'améliorer les résultats de la classification.

1.3.1 Représentation du texte

Après le prétraitement du texte, une étape importante du processus de classification est la représentation du texte. Cette étape consiste à transformer les données textuelles en une forme adaptée à l'analyse et à l'apprentissage automatique. Plusieurs méthodes de représentation du texte sont couramment utilisées, parmi lesquelles on peut citer [4] :

Représentation par sac de mots (Bag-of-Words) :

Cette méthode consiste à représenter chaque document sous la forme d'un vecteur qui compte l'occurrence ou la fréquence des mots présents dans le texte, sans considérer leur ordre ou leur structure grammaticale. Cela permet de capturer les informations de fréquence des mots, mais ne prend pas en compte la séquence des mots.

Représentation avec le n-Gram :

C'est une approche de représentation du texte qui consiste à considérer des séquences contiguës de n éléments (généralement des mots) dans le texte. La méthode n-grammes permet de capturer l'information de contexte local et peut être utilisée en combinaison avec d'autres méthodes de représentation du texte. Elle est utile pour capturer des relations lexicales et syntaxiques spécifiques. Cependant, il est important de prendre en compte la dimensionnalité de l'espace de représentation et d'appliquer des techniques de sélection des caractéristiques pour éviter les problèmes de coût

computationnel et de surapprentissage.

Représentation par Plongement lexical (word embeddings) :

Les plongements de mots sont des représentations vectorielles apprises à partir de grands corpus de textes. Ces plongements capturent les informations sémantiques et syntaxiques des mots, permettant ainsi de représenter le sens des mots et de capturer les relations entre eux. Des méthodes populaires d'apprentissage d'embeddings incluent Word2Vec, GloVe et FastText.

1.3.1.1 Pondération des termes

Dans cette étape, des poids ou des valeurs numériques sont attribués aux termes présents dans le texte afin d'évaluer leur pertinence pour une tâche donnée. Cela se réalise généralement dans le cadre de la représentation vectorielle des mots ou des documents, où chaque terme est représenté par un vecteur numérique dans un espace multidimensionnel. Voici quelques méthodes de représentation du texte :

TF-IDF (Term Frequency-Inverse Document Frequency) :

C'est une mesure qui évalue l'importance d'un terme dans un document par rapport à une collection de documents. Elle associe à chaque mot un poids qui dépend de sa fréquence dans le document et de sa rareté dans le corpus, pour mettre en avant les mots pertinents. Cette mesure est calculée en multipliant la fréquence du terme dans le document (TF) par l'inverse de sa fréquence dans l'ensemble des documents (IDF) [25].

Fréquence normalisée :

Elle consiste à diviser la fréquence brute d'un terme par le nombre total de termes dans le document afin de prendre en compte la longueur du document [25].

Log-likelihood ratio :

Cette méthode évalue la fréquence d'un terme dans un document en comparant la fréquence observée à la fréquence attendue selon un modèle probabiliste [25].

1.3.2 Classification de texte

La classification est un processus indispensable dans le domaine de l'apprentissage automatique, où des étiquettes ou des classes prédéfinies sont attribuées à un ensemble d'observations en se basant sur des caractéristiques spécifiques.

Le processus d'apprentissage et de classification comprend généralement deux étapes principales. La première étape consiste à l'entraînement du modèle, où un algorithme est utilisé

pour apprendre à partir d'un ensemble de données d'entraînement. Cet ensemble de données contient des exemples étiquetés, c'est-à-dire des données avec leurs catégories ou étiquettes correspondantes. Pendant l'entraînement, l'algorithme utilise ces exemples pour découvrir les relations qui permettent de distinguer les différentes catégories [7].

La deuxième étape est celle de la classification proprement dite, où le modèle formé est utilisé pour prédire les étiquettes des nouvelles données non étiquetées. L'algorithme examine les caractéristiques des données et les compare aux connaissances acquises pendant l'entraînement pour déterminer la catégorie appropriée. L'objectif est de généraliser les connaissances apprises pendant l'entraînement pour pouvoir classer avec précision de nouvelles données [7].

Il existe plusieurs algorithmes populaires utilisés en classification, tels que les arbres de décision, les machines à vecteurs de support (SVM), les k-plus proches voisins (KNN) etc... Chacun de ces algorithmes a ses propres caractéristiques, forces et faiblesses, et est adapté à différents types de problèmes de classification [7].

1.3.3 Evaluation des perfrmance

Les mesures d'évaluation jouent un rôle essentiel dans l'analyse des performances des modèles de classification dans le domaine du traitement automatique du langage naturel (TALN). Nous passons en revue certaines mesures de corrélation couramment utilisées pour évaluer les performances des modèles de classification multi-label, à savoir l'accuracy, la hamming loss et le F1-score :

L'accuracy (ou l'exactitude), est une mesure de corrélation classique utilisée pour évaluer la performance globale d'un modèle de classification multi-label. Elle représente le rapport entre le nombre d'instances correctement classées et le nombre total d'instances.Pour la calculer, on divise la somme des vrais positifs et des vrais négatifs par le nombre total d'instances [40].

Hamming loss, est une mesure pertinente pour la classification multi-label, quantifie le degré de désaccord moyen entre les prédictions du modèle et les étiquettes réelles. Elle mesure la fraction moyenne de positions où les prédictions diffèrent des étiquettes réelles. Une hamming loss plus faible indique une meilleure performance du modèle. Pour la calculer, on divise le nombre de positions où les prédictions diffèrent des étiquettes réelles par le nombre total de positions [40].

Le F1-score, est une mesure qui combine la précision et le rappel pour une classe spécifique. Il est couramment utilisé dans la classification multi-label pour évaluer la performance

de chaque classe individuellement. Le F1-score est calculé en prenant la moyenne harmonique de la précision et du rappel. La précision mesure la proportion de vrais positifs parmi les instances prédites comme positives, tandis que le rappel évalue la proportion de vrais positifs parmi les instances réellement positives [40].

Il existe d'autres mesures d'évaluation, mais nous avons choisi l'accuracy, la hamming loss et le F1-score car elles fournissent des indicateurs essentiels pour évaluer les performances des modèles de classification multi-label dans le domaine du TALN.

1.4 Classification automatique du texte arabe

Bien que cette langue soit très connue et largement parlée dans le monde entier, ses ressources linguistiques utilisables dans le domaine du traitement automatique de la langue ne connaissent pas le même succès. Comparé à d'autres langues, l'arabe est une langue très pauvre en termes d'outils de traitement, même les plus basiques d'entre elles, ainsi que le nombre de corpus ouverts au public pour les utilisations académiques. Ce qui la rend parmi les langues dites « pauvres » dans le domaine. Cela est dû au fait que cette langue, contrairement à l'anglais par exemple, est une langue très complexe et qui ne manque pas d'ambiguïtés.

1.4.1 Complexité de la langue arabe

Dans la recherche d'étiquetage grammatical les auteurs ont pu déceler par exemple que le manque total de voyelles est un facteur qui rend un texte arabe très difficile à traiter car 74% de ses mots sont ambigus rien qu'à cause de ce phénomène à lui seul, notons aussi ce qui suit :

L'absence des voyelles :

L'arabe est une langue qui se base sur le « tachkile » qui est une sorte de signes (اَ اُ اِ اً اٌ اٍ) comme le montre la figure 1.1, pour remplacer les voyelles utilisées par d'autres langues. Donc pour un mot qui peut être lu de plusieurs façons comme le mot "كتب" qui a 7 façons d'écrire, avec un sens différent pour chacune des façon, « katab كَتَبَ : a écrit », « kutiba كُتِبَ : a été écrit » ou encore « kutub كُتُبْ : livres ». Un programme qui fait la différence entre ces formes est très difficile à concevoir [36].

Fatha(فتحة)	Kasra(كسرة)	Dama(ضمة)	Sukun(سكون)	Chada(شدة)	Tanwin(تنوين)		
◌َ	◌ِ	◌ُ	◌ْ	◌ّ	◌ً	◌ٌ	◌ٍ

TABLE 1.1 – Signes diacritiques [36].

Vocabulaire très large :

Le vocabulaire de la langue arabe est très large, au point ou le mot «حب» peut avoir plusieurs autres mots qui l'expriment.

La gémination :

Représente ou doublement d'une lettre pour marquer sa prononciation, pour le français par exemple, le mot grammaire à le « m » en double mais qui ne se voit pas lors de la prononciation du mot, par contre « kattaba كتّب » sera bien plus visible et ainsi affectera le sens et le type du mot en question, en effet, « kattaba » signifie « il a fait écrire » alors que « kataba » avec un seul t veut dire « il a écrit » [36].

La classification du texte en arabe peut nécessiter des adaptations spécifiques en raison des caractéristiques particulières de cette langue telles que la présence de formes verbales complexes, la variation des mots en fonction de leur genre et de leur nombre, ainsi que la présence fréquente de mots racines avec de multiples dérivations. Ces spécificités peuvent nécessiter des techniques de prétraitement, de segmentation et de normalisation spécifiques pour une classification précise du texte arabe. comme exemple des techniques de prétraitement spécifique pour une classification du texte arabe il est possible de citer :

La normalisation des caractères :

L'arabe a des variantes de caractères qui peuvent être normalisées pour faciliter le traitement, par exemple, en utilisant des formes standardisées pour les lettres [26].

La suppression des diacritiques :

Les diacritiques sont les signes qui modifient la prononciation et la signification des lettres arabes. La suppression des diacritiques peut simplifier le traitement et la comparaison des mots [26].

1.5 classification multi-étiquettes

L'une des méthodes viables pour classifier des données est la classification multiclasse qui est une tâche d'apprentissage automatique qui a pour but d'assigner des classes à des données ou des documents textuels. Cela étant dit, certaines instances peuvent appartenir à

plusieurs classes en même temps, chose que la classification multiclasse ne peut faire. Dans ce cas, la classification multi-étiquettes est la parfaite solution pour classer des données appartenant à plusieurs classes en même temps. Contrairement à la classification binaire ou à la classification multi-classe où chaque échantillon ne peut être associé qu'à une seule étiquette, la classification multi-étiquette (multi-label classification ou MLC) est un sous-domaine de l'apprentissage automatique où chaque échantillon de données est assigné à plusieurs étiquettes en même temps. Il est généralement insuffisant de classer chaque instance sous une seule étiquette, car il existe plusieurs étiquettes qui pourraient convenir pour décrire son contenu simultanément [2].

1.6 Les approches de classification multi-étiquettes

La résolution de la tâche de classification multi-label est plus complexe comparée à la classification binaire ou multi-classe, car il faut associer les bonnes étiquettes à chaque donnée et pouvoir traiter ces dernières de façon efficace. Il existe plusieurs méthodes pour résoudre cette tâche, la figure 1.1 représente la taxonomie des différentes méthodes de la classification multi-étiquettes [2].

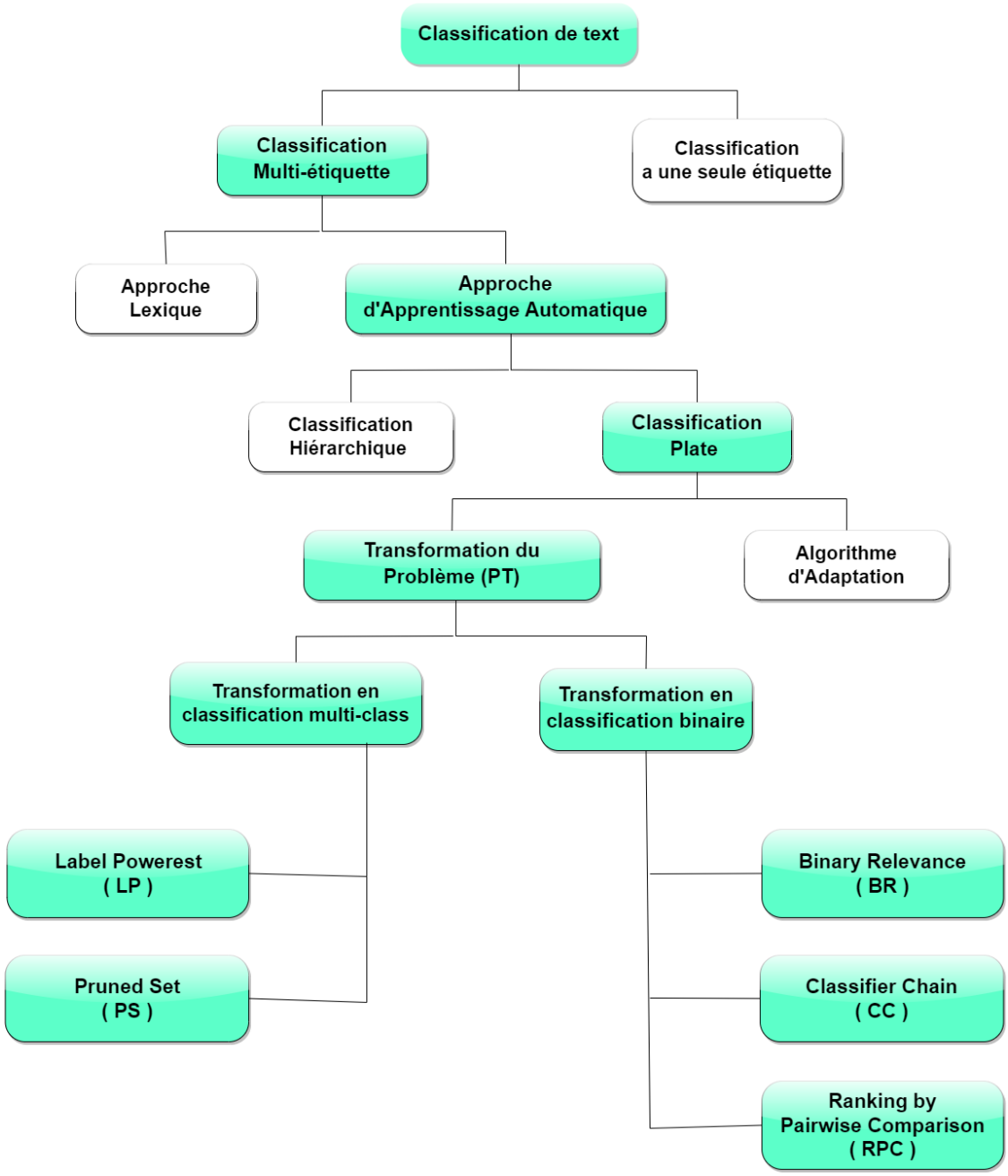

FIGURE 1.1 – Taxonomie des méthodes de la classification multi-étiquettes.

Comme le montre la figure 1.1 ci-dessus, la classification multi-étiquette connaît deux approches différentes. D'abord l'approche lexique qui repose sur l'utilisation de lexiques ou de ressources lexicales pour améliorer la performance des modèles de classification multi-étiquettes. Cette approche exploite les informations sémantiques contenues dans ces lexiques pour enrichir la représentation des données d'entrée et faciliter la prédiction des étiquettes associées [2].

Quand a la deuxième approche c'est une approche d'apprentissage automatique qui est composée de :

1. **La classification hiérarchique :**

   HMC (Hierarchical Multi-label Classification) qui est considérée comme une extension de la classification multi-étiquette où une structure hiérarchique est appliquée sur le multi-label. Dans HMC, une instance est associée avec plusieurs étiquettes en même temps et ces étiquettes sont classifiées selon une hiérarchie, cette instance doit respecter la contrainte de hiérarchie [2].

2. **La classification plate :**

   Également connue sous le nom de classification à plat ou de classification non hiérarchique, est une méthode de classification des données dans laquelle les éléments sont assignés directement à des classes ou des catégories sans établir de relations hiérarchiques entre elles [2].

En multi-étiquette, la classification plate est divisée en deux techniques à savoir la technique d'adaptation d'algorithme et la technique de transformation de problème. Dans notre étude, nous nous intéressons principalement aux méthodes de transformation de problème, mais on ne manquera pas de définir brièvement la méthode d'adaptation d'algorithme [21].

2.1. La transformation de problème :

Le principe de ces méthodes est de transformer le problème multi-étiquettes en un groupe de problèmes à étiquette unique, puis utiliser les algorithmes d'apprentissage classique pour effectuer la classification. On peut distinguer deux techniques de de transformation de problème : la première repose sur la transformation en classification binaire, tandis que la deuxième est basée sur la transformation en classification multi-classe [21].

Dans les deux techniques, la tâche de classification est basée sur des modèles de classification à étiquette unique. Cependant, les ensembles similaires d'étiquettes sont regroupés en une classe distincte dans le cas de la classification multi-classe. Ainsi, chaque ensemble d'étiquettes similaires trouvées dans l'ensemble de données d'apprentissage est considéré comme une nouvelle classe :

1. Les techniques de transformation de problèmes multi-étiquettes en problèmes binaires :

   - **Binary Relevance (BR) :**

     La pertinence binaire (Binary Relevance) est la méthode la plus simple pour la transformation de problèmes de classification multi-étiquettes en classification binaire. Le principe de BR consiste à transformer le problème multi-étiquette en classification binaire en attribuant un classifieur binaire à chaque

classe. De ce fait, N problèmes de classification binaire sont déduits (ou N représente le nombre de classes) et l'appartenance d'une instance à chaque classe sera prédite d'une manière indépendante. Les instances des données multi-étiquettes d'origine sont incluses dans chaque donnée à étiquette unique et elles sont prédites avec une étiquette positive si elles ont l'étiquette existante, sinon elles sont prédites avec une étiquette négative. Une nouvelle instance se verra attribuer les étiquettes positives prédites par les N classificateurs binaires [13] ;
Prenons l'exemple représenté dans la figure 1.2, ou X sont les instance et Y les classes :

X	Y_1	Y_2	Y_3	Y_4
$x^{(1)}$	0	1	1	0
$x^{(2)}$	1	0	0	0
$x^{(3)}$	0	1	0	0
$x^{(4)}$	1	0	0	1
$x^{(5)}$	0	0	0	1

FIGURE 1.2 – Représentation du Principe Binary Relevance [15].

Dans Binary Relevance, ce problème est divisé en 4 problèmes à une seule étiquette, on aura donc 4 classifieurs (1 pour chaque classe) comme le montre la figure 1.3 :

X	Y_1
$x^{(1)}$	0
$x^{(2)}$	1
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	0

X	Y_2
$x^{(1)}$	1
$x^{(2)}$	0
$x^{(3)}$	1
$x^{(4)}$	0
$x^{(5)}$	0

X	Y_3
$x^{(1)}$	1
$x^{(2)}$	0
$x^{(3)}$	0
$x^{(4)}$	0
$x^{(5)}$	0

X	Y_4
$x^{(1)}$	0
$x^{(2)}$	0
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	1

FIGURE 1.3 – Représentation du Principe Binary Relevance [15].

Parmi les avantages de BR, la complexité de calcul raisonnable, cela étant dit, elle ne considérer pas la dépendance entre les classes [13].

- **Classifier Chain (CC) :**

  La méthode de la Chaîne de Classification (Classifier Chain), est une amélioration de la méthode BR qui transforme également le problème d'apprentissage multi-label des problèmes mono-label. Contrairement à la méthode BR, cette méthode chaine de classification tient compte des dépendances entre les classes et de ce fait elle résout l'inconvénient de BR. Le principe de CC est de transformer le problème multi-étiquette (Multi-Label) en une chaîne de

problèmes de classification binaire où chaque résultat de classification (prédiction) est considéré comme attribut futur dans le prochain classifieur, ce qui veut dire que les informations d'étiquette sont transmisent entre les étiquettes, ce qui nous permet de tenir compte des dépendances entre les classes [14].
Prenons l'exemple de la figure 1.4, ou x1, x2, x3 représentent les instances et y1, y2, y3, y4 représentent les classes :

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

FIGURE 1.4 – Probleme multi-étiquettes 02 [15].

Avec Classifier Chain ,ce problème serait transformé en 4 problèmes distincts à étiquette unique, comme illustré ci-dessous dans la figure 1.5. La zone colorée en jaune représente l'espace d'entrée et la partie blanche notre variable cible :

X	y1
x1	0
x2	1
x3	0

X	y1	y2
x1	0	1
x2	1	0
x3	0	1

X	y1	y2	y3
x1	0	1	1
x2	1	0	0
x3	0	1	0

X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

FIGURE 1.5 – Représentation du Principe Classifier Chain [15].

L'un des principaux avantages de CC c'est qu'elle considère la dépendance entre les étiquettes pour obtenir des performances prédictives élevées tout en maintenant une complexité de calcul raisonnable (celle du BR), et pour les inconvénients on peut citer la propagation d'erreur, car en effet en cas de mauvaise prédiction d'un classifieur, cela peut affecter la prédiction des étiquettes suivantes, c'est la propriété de chainage qui provoque cette propagation [14].

- **Ranking by Pairwise Comparison (RPC) :**

  Classement par comparaison par paires (Ranking by Pairwise Comparison) est un autre type de méthode de transformation de problème, c'est une extension de la méthode originale de comparaison par paires (PC Pair Comparison).

  RPC est divisé en 2 étapes : dans la première, l'ensemble de données multi-étiquettes avec N étiquettes est transformé en N(N-1)/2 problèmes de classification binaire à une seule étiquette, on associe a chaque paire d'étiquettes un classificateur binaire en couvrant toutes les paires d'étiquettes. Chaque problème de classification binaire unique implique les instances du problème multi-étiquettes , qui sont affectées à une seule des deux classes seulement [2] ;

Tout d'abord, dans la tâche de prédiction d'étiquettes de chaque classificateur, le classificateur effectue une comparaison entre chaque paire d'étiquettes. Par exemple, s'il compare entre l'étiquette 1 et l'étiquette 2, l'étiquette prédite résultante sera soit 0 ou 1 selon l'équation (1).
Dans cette équation 'y' représente l'étiquette prédite :

$$y = \begin{cases} 1, \text{ si étiquette } 1 > \text{étiquette } 2 \\ 0, \text{ si étiquette } 2 > \text{étiquette } 1 \end{cases} \tag{1.1}$$

Dans la deuxième étape, les étiquettes prédites sont classées à l'aide d'une procédure de classement telle qu'une généralisation de la stratégie de vote où toutes les étiquettes sont classées sur la base de la somme évaluée des votes pondérés. Pour classer une nouvelle instance à l'aide de la méthode RPC, chaque classificateur binaire est invoqué et vote pour l'une des deux étiquettes. Après prédiction des étiquettes par tous les classifieurs, le classement des étiquettes est obtenu selon la somme des votes de chaque label.

Le principal inconvénient du RPC est la nécessité d'interroger tous les classifieurs générés au moment de la classification. De plus, une complexité quadratique (n) de RPC le rend très sensible au grand nombre d'étiquettes et il est généralement difficile de traiter de gros problèmes [2].

2. Les techniques de transformation de problèmes multi-étiquettes en problèmes binaires :

- **Label Powerset :**

  Label powerset (LP) est la méthode la plus simple pour la transformation de problèmes de classification multi-étiquettes en problemes de classification multi-classe. Le principe de LP est de transformer le problème MLC en un problème de classification multi-classes puis percevoir chaque ensemble d'étiquettes distinct dans les données d'apprentissage comme une nouvelle classe d'une tâche de classification multi-classes. On utilise un classifieur multi-classe pour pouvoir classer une nouvelle instance en lui associant la classe la plus vraisemblable parmi plusieurs autres nouvelles classes (rappelons que dans LP une classe représente un ensemble d'étiquette distinct), après attribution d'une classe à cette instance, elle sera reversée à l'ensemble d'étiquettes initiales associées.

  LP a quelques inconvénients, le premier est qu'il a une complexité de calcul qui augmente de façon exponentielle avec le nombre d'étiquettes et c'est la rai-

son pour laquelle LP se détériore rapidement pour des ensembles d'étiquettes plus grands et rend le travail du classifieur plus difficile, c'est pourquoi LP n'est recommandé que pour les ensembles de données qui ont un petit nombre de classes distinctes. De plus, on peut avoir de nouvelles classes qui ont un petit nombre d'instances à cause des combinaisons d'étiquettes peu fréquentes ce qui crée un problème de déséquilibre de classes [23].

- **Pruned Set (PS) :**

  Pruned set (PS) est une variante de la méthode LP, le principe de PS est le même que celui de LP, la seule différence est que PS nous permet d'éliminer les combinaisons d'étiquettes qui sont peu fréquentes et ça en permettant à l'utilisateur de définir un seuil, et toute classe ayant une fréquence d'occurrence inférieur à ce seuil sera éliminée [16].
  De ce fait, la complexité qui cause un problème dans la méthode LP est réduite dans la méthode PS, par conséquent, il réduit la complexité LP en ne conservant que l'ensemble d'étiquettes qui se produisent plus fréquemment en le comparant au seuil [16].

2.2. La technique d'adaptation d'algorithmes :

Les algorithmes classiques de classification à étiquette unique ne peuvent pas traiter directement le problème MLC, donc une technique d'adaptation d'algorithme a été développée pour résoudre le problème MLC. Elle consiste à adapter l'algorithme de classification à étiquette unique pour pouvoir traiter les problèmes MLC. Beaucoup de classifieurs classiques à étiquette unique ont été adaptés pour traiter les problèmes multi-étiquettes, on peut citer par exemple : l'algorithme SVM qui a été adapté à Rank SVM , ou bien l'algorithme KNN classique qui a été adapté lui aussi a ML-KNN, ou encore l'algorithme d'arbre de décision multi-étiquettes (ML-DT) a été développé en adaptant l'algorithme d'arbre de décision C4.5 [22].

1.7 Travaux connexes

La classification multi-étiquettes (MLC Multi-Label Classification) est utilisée dans notre vie quotidienne pour résoudre divers problèmes comme par exemple la classification d'émail ou la classification de textes. La majorité des études et recherches sont consacrées à la classification multi-étiquettes en langue anglaise, contrairement aux recherches en langue arabe qui sont jugées pauvres et insuffisantes. Certains des travaux déjà réalisés en langue arabe sont représentés dans le tableau 1.2 suivant :

Référence	Corpus	Representation	Classification	Résultats
(Nizar &al, 2015) [37]	BBC (7000 articles)	TF-IDF (Term Frequency-Inverse Document Frequency)	Support Vector Machine	-Accuracy : 65% -Perte de Hamming : 0.012%
(Shehab &al, 2016) [22]	CNN (11000 articles)	TF-IDF (Term Frequency-Inverse Document Frequency)	Random Forest	-accuracy : 90% -Perte de Hamming : 0.2%
(Taha, & Tiun, 2016) [20]	BBC (10000 articles)	TF-IDF (Term Frequency-Inverse Document Frequency)	Ensemble de classifieurs KNN	- Précision : 97% - F1-score : 85%
(Al-salemi &al, 2019) [19]	RTA news (72500 articles)	TF-IDF (Term Frequency-Inverse Document Frequency)	BR avec KNN.	- Macro précision : 83% - Micro précision : 85%

TABLE 1.2 – Représentation des travaux connexes.

1.8 Conclusion

Ce chapitre nous a permis d'acquérir une compréhension approfondie de l'état actuel de la classification à une seule étiquette et de la classification multi-étiquette. Ces connaissances serviront de base solide pour la suite de notre étude, où nous aborderons la proposition d'une méthode innovante pour la classification multi-étiquette en langue arabe en les relation entre les étiquettes.

Chapitre 2

Conception

2.1 Introduction

Dans ce chapitre, nous allons présenter en détail notre approche qui s'inspire des algorithmes de classification multi-étiquettes qui existent, BR et CC, en mettant l'accent sur la manière dont nous traitons la corrélation entre les étiquettes. Nous présenterons les différentes étapes de conception, les algorithmes utilisés et les mesures de corrélation. Mais d'abord, nous allons parler du corpus de la langue arabe auquel nous aurons recours à savoir BBC (British Broadcasting Corporation). Ce corpus a été nettoyé et traité pour faire l'objet de notre travail.

2.2 Contribution de notre travail

L'apprentissage multi-étiquettes est une approche étudiée pour traiter des problèmes où plusieurs étiquettes sont associées simultanément aux échantillons donnés, et dans de nombreuses tâches de classification du texte, il est courant qu'il existe une interdépendance entre les étiquettes des documents textuels, et comme on l'a vu précédemment dans la section **"Travaux connexes (1.7)"** du chapitre 01, la majorité des travaux antérieurs proposés pour la classification multi-étiquettes du texte arabe ont négligé cet aspect important et n'ont pas exploité cette corrélation entre les étiquettes pour améliorer la performance de la classification.

Notre travail se concentre sur un objectif ambitieux : améliorer la précision de la classification multi-étiquettes du texte arabe en exploitant les relations et les similarités entre les différentes catégories. Pour atteindre cet objectif, nous proposons un système novateur basé sur la corrélation entre les étiquettes. En analysant les liens subtils entre les catégories prédéfinies, nous cherchons à maximiser la pertinence et la cohérence des résultats de classification. Notre approche vise à mieux capturer la richesse et la complexité du langage arabe, offrant ainsi une solution plus efficace et précise pour l'analyse et l'organisation des textes

dans un contexte multi-étiquettes. Notre recherche ouvre ainsi de nouvelles perspectives dans le domaine de la classification du texte arabe.

2.3 Aperçu sur la corrélation

La corrélation est un terme statistique qui exprime à quel point deux variables sont étroitement liées de manière linéaire (ce qui signifie qu'elles évoluent ensemble à un taux constant). C'est une façon courante de décrire des relations simples sans établir de relation de cause à effet, elle peut être évaluée en utilisant diverses techniques statistiques, telles que le test du chi-carré, les mesures de corrélation de Pearson ou de Spearman, etc. Ces méthodes permettent de déterminer si une corrélation positive, négative ou aucune corrélation significative existe entre les variables [17].

Comme le montre la figure 2.1 la corrélation positive indique que lorsque la valeur d'une variable augmente, la valeur de l'autre variable augmente également, et vice versa. Cela suggère une relation directe et cohérente entre les deux variables. Une corrélation positive est représentée par un coefficient de corrélation compris entre 0 et 1, où une valeur plus proche de 1 indique une corrélation plus forte [18].

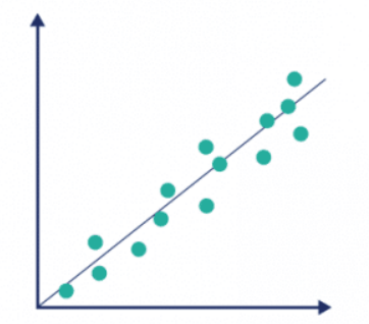

FIGURE 2.1 – Forte Corrélation positive [18].

En revanche, la figure 2.2 représente une corrélation négative indiquant que lorsque la valeur d'une variable augmente, la valeur de l'autre variable diminue, et vice versa. Cela indique une relation inverse ou opposée entre les deux variables. Une corrélation négative est représentée par un coefficient de corrélation compris entre -1 et 0, où une valeur plus proche de -1 indique une corrélation plus forte [18].

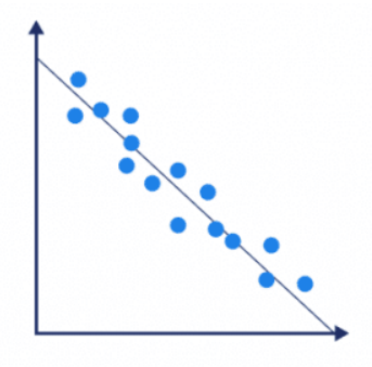

FIGURE 2.2 – Forte corrélation négative [18].

Enfin, la figure 2.3 représente une corrélation neutre (ou aucune corrélation).Une absence de corrélation signifie qu'il n'y a pas de relation apparente entre les deux variables. Les variations d'une variable ne sont pas liées de manière significative aux variations de l'autre variable. Une corrélation neutre est représentée par un coefficient de corrélation proche de zéro, indiquant une absence de corrélation linéaire entre les variables [18].

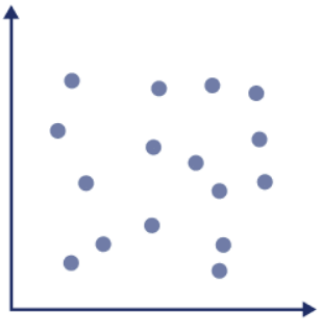

FIGURE 2.3 – Corrélation neutre [18].

Dans notre approche on utilise la mesure de corrélation de Pearson, qui est une méthode statistique utilisée pour évaluer la corrélation linéaire entre deux variables continues. Elle mesure la force et la direction de la relation linéaire entre les deux variables [17].

Le coefficient de corrélation de Pearson est compris entre -1 et 1. Comme indiqué auparavant, un coefficient de corrélation de +1 indique une corrélation positive parfaite, ce qui signifie que les variables sont linéairement liées et évoluent ensemble de manière constante. Un coefficient de -1 indique une corrélation négative parfaite, où les variables sont linéairement liées mais évoluent en sens inverse. Un coefficient de corrélation proche de 0 indique une faible corrélation linéaire ou une absence de relation linéaire entre les variables [17].

2.4 Choix du Dataset

Pour tester notre approche, nous avons besoin d'un ensemble de données en langue arabe. Cependant, nous avons été confrontés à une limitation dans notre choix vu le manque des

ensembles de données en langue arabe. En effet, le choix du corpus multi-etiquettes pour la classification de texte arabe pose un vrai défi en raison du du petit nombre d'ensemble de données disponibles.

Pour réaliser notre travail, nous avons choisi le corpus standard proposé par Ahmed et al. [21], qui est composé d'articles de presse collectés en ligne à partir du site de la BBC en arabe. Ce dataset consiste en un ensemble de plus de 5000 articles et de 57 catégories parmi lesquelles on peut citer : "علوم", "سياسة", "تجارة وأعمال", "ثقافة وفنون" et "رياضة". Chaque article est associé à un ensemble de sujets connexes, qui sont considérés comme des étiquettes pour l'article, permettant ainsi à un article d'appartenir à plusieurs classes en même temps (maximum 5 classes).

2.5 Architecture globale du système proposé

Le choix et la complexité d'une architecture peuvent affecter le fonctionnement et les performances globales d'un système de classification. Dans notre travail, nous avons proposé une architecture qui répond aux exigences du corpus choisi et qui vise à assurer de bonnes performances. Notre architecture est basée sur trois phases :

1. **Prétraitement des articles news :**
   Le but de cette phase est de normaliser et standardiser le format des mots des articles en utilisant des techniques de nettoyage du texte, suppression de mots vides et de racinisation.
2. **Représentation du texte des articles :**
   La technique TF-IDF est adoptée à ce stade pour représenter chaque article d'une manière numérique.
3. **Classification multi-étiquettes :**
   Un nouveau système de classification multi-étiquettes est exploité à cette étape afin de classifier les articles en langue arabe se basant sur l'utilisation de la corrélation entre les étiquettes.

Une description détaillée de chaque étape de notre approche est représentée dans les sections suivantes. Pour faciliter la compréhension, un schéma récapitulant ces différents points est illustré dans la figure 2.4 :

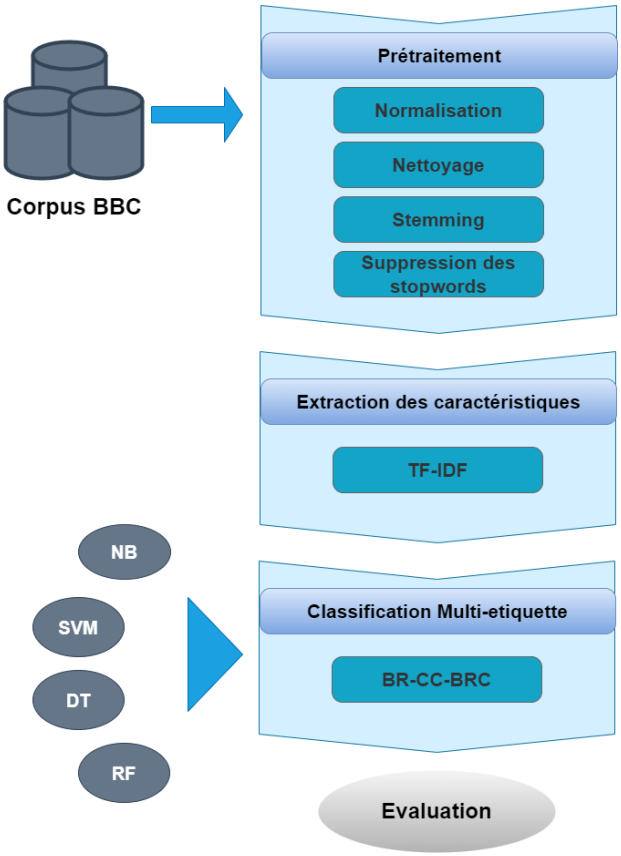

FIGURE 2.4 – Architecture du système proposé.

2.6 Prétraitement du corpus

Le prétraitement de donnée est sans doute une étape inévitable si on veut travailler sur n'importe quel corpus, Pour rendre notre corpus BBC plus adapté à notre approche, nous avons effectué un nettoyage en éliminant les éléments indésirables tels que la ponctuation, les signes diacritiques et les mots outils (stop words). Nous avons commencé par supprimer les colonnes inutiles, telles que l'ID, l'URL et le titre. Ensuite, nous avons éliminé les lignes vides et les duplications pour éviter toute redondance. Nous avons ensuite procédé aux étapes suivantes dans l'ordre dans lequel elles sont présentées :

1. **Suppression des diacritiques :**

   Voici un exemple avant la suppression des diacritiques : اللُّغَةُ العَرَبِيَّةُ جَمِيلَةٌ وَمُفِيدَةٌ.

   Voici le résultat apres la suppression des diacritiques : اللغة العربية جميلة ومفيدة.

2. **Suppression des URL :**
   Voici un exemple avant la suppression des URL :
   : www.example.com

   Voici le résultat apres la suppression des URL : قم بزيارة الموقع الإلكتروني التالي للحصول على مزيد من المعلومات.

3. **Suppression des espaces supplémentaires :**
   Voici un exemple avant la suppression des espaces supplementaires : ال تق دم ا ل ت ع ل ي مي

   Voici le résultat apres la suppression des espaces suppelementaires : التقدم التعليمي

4. **Suppression des mots non arabes :**
   Voici un exemple avant la suppression des mots non arabes : أحب القهوة وأستمتع بتناولها et

   Voici le résultat apres la suppression des mots non arabes : أحب القهوة و أستمتع بتناولها

5. **Suppression des symboles non arabes :**
   Voici un exemple avant la suppression des symboles non arabes : أحب القهوة / ^ #

   Voici le résultat apres la suppression des symboles non arabes : أحب القهوة

6. **Suppression des nombres :**
   Voici un exemple avant la suppression des nombres : أحب القهوة 4

   Voici le résultat apres la suppression des nombres non arabes : أحب القهوة

7. **Normalisation du texte arabe :**
   En remplaçant certaines variantes de lettres par leurs formes standardisées, comme remplacer les ["ة","آ","إ","أ"] par des ["ه","ا","ا","ا"].

8. **Suppression des mots vides (stop words) :**

   Tels que [إلى، من، على ، و] et d'autres.

9. **Racinisation (stemming) :**
   Pour ramener les mots à leur forme canonique en éliminant les préfixes et les suffixes, nous prenons l'exemple suivant :
   Avant la racinisation : "الاستثمارات"

Après la racinisation : "الاستثمارات"

2.7 Représentation du texte

La vectorisation des données vient juste après la phase de prétraitement, elle permet de transformer les textes en vecteurs numériques exploitables pour faire la classification de document. Dans ce chapitre, nous nous focalisons spécifiquement sur l'application de la méthode TF-IDF (Term Frequency-Inverse Document Frequency) pour la vectorisation des données textuelles.

2.7.1 Présentation de TF-IDF

Le TF-IDF est une mesure statistique qui évalue l'importance d'un terme dans un document par rapport à une collection de documents. Il est basé sur deux principaux concepts :

1. **La fréquence du terme (TF) :** qui désigne le nombre d'apparitions d'un terme donné dans un document.Généralement, il est pondéré pour tenir compte de la longueur du document.
2. **La fréquence inverse du document (IDF) :** qui mesure l'importance d'un terme en fonction de son occurrence dans la collection de documents. Les termes fréquents dans l'ensemble des documents ont une valeur IDF faible, tandis que ceux qui apparaissent rarement ont une valeur IDF élevée.

2.7.2 Calcul des matrices TF-IDF

Après avoir nettoyé notre corpus, on calcul les matrices TF-IDF, ou on va construire une matrice termes-documents où chaque ligne représente un document et chaque colonne représente un terme. Les valeurs dans cette matrice sont calculées en utilisant les formules de TF et IDF suivantes :

$$W_{i,j} = TF(tj, Mi).IDF(i, j) \tag{2.1}$$

$$TF(tj, Mj) = \frac{freq(tj, Mi)}{\sum_{j=1}^{m}(freq(tj, Mi))} \tag{2.2}$$

$$IDF(tj) = logTDF(tj) \tag{2.3}$$

où :

- T est la taille du corpus.
- DF(tj) est le nombre de messages contenant le terme tj dans le corpus.
- $freq$(tj,Mi) est le nombre de fois où le mot j est apparu dans le message i.
- La somme $\sum_{j=1}^{m}(freq$(tj,Mi)) donne le nombre de mots dans le message i.

Par la suite, les vecteurs TF-IDF peuvent être utilisés pour alimenter des algorithmes d'apprentissage automatique.

2.8 Classification multi-étiquette

Le système de classification multi-étiquettes du texte arabe proposé comprend quatre phases principales : Calcul de la matrice de corrélation, Génération des listes des labels corrélés, Phase d'entraînement et Phase de classification. La figure 2.5 présente le schéma global du système proposé. Les sections suivantes abordent en détail chaque étape du système proposé.

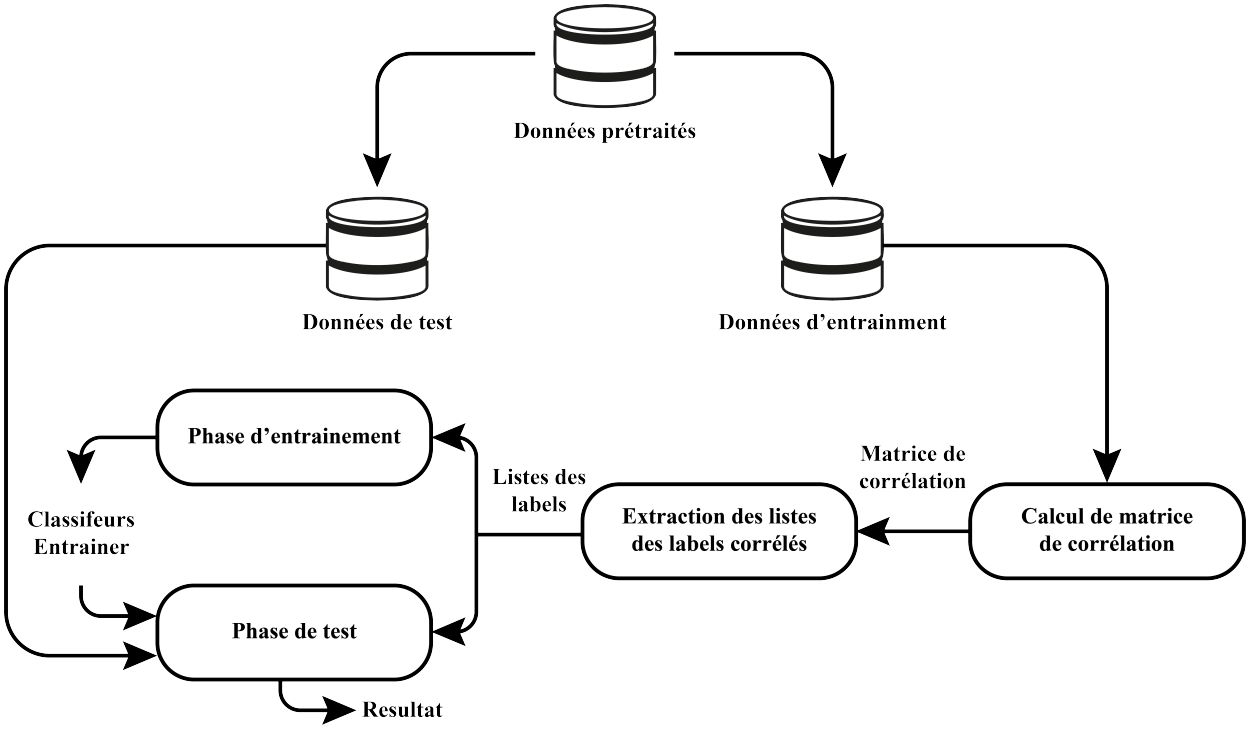

FIGURE 2.5 – Schéma global de l'approche proposée

2.8.1 Matrice de corrélation

La matrice de corrélation analyse les dépendances entre les différentes étiquettes. Cette matrice est une représentation tabulaire qui montre les coefficients de corrélation entre les variables, ainsi elle nous permet d'identifier les groupes d'étiquettes corrélées pour chaque label. Chaque cellule de la matrice indique la fréquence d'apparition des paires d'étiquettes X et Y dans les articles.

Dans notre approche on utilise la mesure de corrélation de Pearson, illustrée dans la figure 2.6, qui est une méthode statistique utilisée pour évaluer la corrélation linéaire entre deux variables continues. Elle mesure la force et la direction de la relation linéaire entre les deux variables (dans notre étude entre 2 étiquettes) [17].

	آسيا	أستراليا	أفغانستان	أمريكا الشمالية	أمريكا اللاتينية	أوروبا
آسيا	0.000000	0.108570	-0.016671	-0.019659	-0.026447	-0.031726
أستراليا	0.108570	0.000000	-0.005421	-0.004108	-0.005526	-0.012087
أفغانستان	-0.016671	-0.005421	0.000000	-0.008973	-0.012071	-0.008136
أمريكا الشمالية	-0.019659	-0.004108	-0.008973	0.000000	-0.009146	-0.020004
أمريكا اللاتينية	-0.026447	-0.005526	-0.012071	-0.009146	0.000000	-0.026912
أوروبا	-0.031726	-0.012087	-0.008136	-0.020004	-0.026912	0.000000

FIGURE 2.6 – Échantillon de la matrice de corrélation.

Le coefficient de corrélation de Pearson est compris entre -1 et 1. Un coefficient de corrélation égale à +1 indique une corrélation positive parfaite, ce qui signifie que les variables sont linéairement liées et évoluent ensemble de manière constante. Un coefficient égale à -1 indique une corrélation négative parfaite, où les variables sont linéairement liées mais évoluent en sens inverse. Un coefficient de corrélation proche de 0 indique une faible corrélation linéaire ou une absence de relation linéaire entre les variables [17].

2.8.2 Génération des listes d'étiquettes corrélés

En analysant la matrice de corrélation mentionnée précédemment, nous pouvons générer des listes d'étiquettes qui sont corrélées les unes aux autres. Pour décider du degré de corrélation à prendre en compte, nous avons établi un seuil fixé a 0.5. Tout coefficient supérieur à notre seuil indique l'existence d'une corrélation entre ces deux labels. Un coefficient de corrélation inférieur ou égale à notre seuil indique une faible corrélation entre les deux classes, c'est pourquoi nous ne prenons en compte que les coefficients supérieurs au seuil. La figure 2.7 montre un exemple d'une liste d'étiquettes corrélés générée :

```
la liste des étiquettes corrélées: [مجلة العلوم والتكنولوجيا, مجلة الاقتصاد, المجلة ]
```

FIGURE 2.7 – Liste des éttiquettes corrélées.

2.8.3 Phase d'entraînement

Dans notre approche, nous nous concentrons sur l'utilisation de deux approches de classification multi-étiquette : Binary Relevance (BR) et Classifier Chains (CC), tout en prenant en compte la corrélation entre les étiquettes.

Pour profiter des avantages de ces deux approches nous avons d'abord appliqué une classification respectant le principe de BR pour les classes non corrélées, puis par la suite appliquer une classification respectant le principe CC sur les listes de classes corrélées, de ce fait on a

garder une complexité de calcul assez raisonnable en utilisant le principe de BR, mais aussi on a pu prendre en considération les dépendances existantes entre les classes en incorporant le mode de fonctionnement de CC sur les listes de classes corrélées qu'on a obtenu comme on l'a expliqué dans la section **"Génération des listes de classes corrélées 2.8.3"**, la figure 2.8 exemplifie le principe de notre approche :

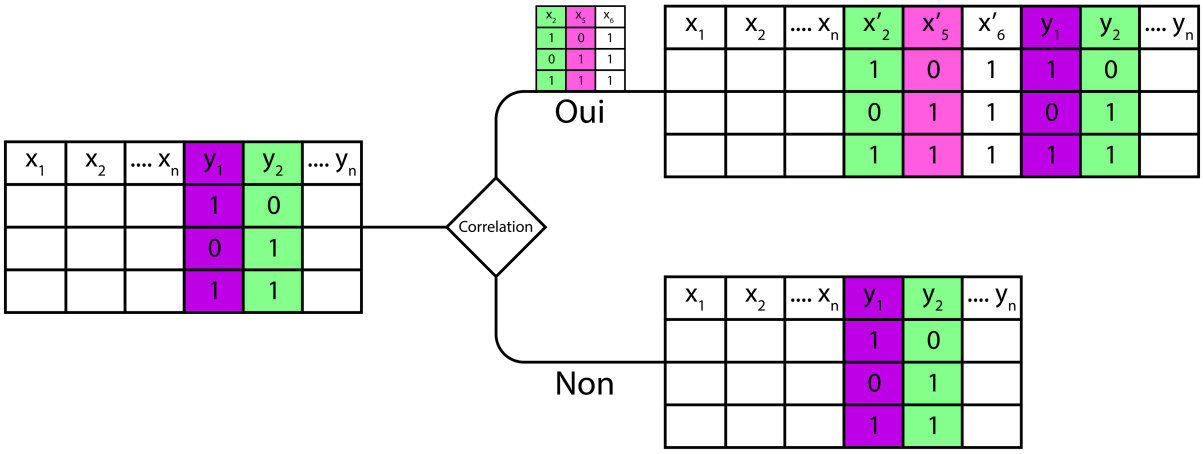

FIGURE 2.8 – Representation du princepe BRC.

Voici notre algorithme en langage humain pour simplifier la compréhension de notre approche :

Algorithme 1 : Fonction d'entrainement

Fonction fit(X, y)

Entrées :

X : tableau-like ou matrice creuse, shape = [n_samples, n_features]
Les échantillons d entrée d entraînement.

y : tableau-like, shape = [n_samples, n_labels]
Les valeurs cibles (étiquettes de classe) en tant qu entiers ou chaînes de caractères.

Variables locales :

corr_mat : Matrice de corrélation de Pearson entre les classes (initialisée à None)

seuil : Seuil de corrélation pour la sélection des features (initialisé à None)

models : Liste des classifieurs individuels pour chaque classe (initialisée à une liste vide)

dict_labels : Dictionnaire pour stocker la correspondance entre le label et l indice (initialisé à un dictionnaire vide)

col_f : Liste des noms des nouvelles features ajoutées (initialisée à une liste vide)

Début :

```
# Vérifier si la matrice de corrélation existe
Si corr_mat !existe Alors
    corr_mat = y.correlation()
Si seuil !existe Alors
    seuil = corr_mat.moyenne()
Pour label dans liste_des_labels(y)
    dict_labels[label] = indice
    corr_class = Sélectionner les classes corrélées avec 'label' à partir
                 de corr_mat
    # Si des classes corrélées sont trouvées
    Si longueur de corr_class > 0 Alors
        # Pour chaque classe corrélée
        Pour cl dans liste_des_classes_corrélées
            Affecter les valeurs de la colonne 'cl' de y à la colonne 'cl' de X
col_f = Sauvegarder ici les noms des nouvelles features ajoutées
Pour label dans liste_des_labels(y)
    # Ajuster le modèle de base sur les données
    base_model.fit(X, y)
    Ajouter base_model à la liste des classifieurs individuels
```

Sorties : liste des classifieurs entrainées

Fin. :

Fin de la fonction

2.8.4 Phase de prédiction

La phase de prédiction vise à utiliser le modèle préalablement entraîné sur de nouveaux textes ou données non vus auparavant pour effectuer des prédictions. Dans notre étude il s'agit de prévoir des étiquettes. Lors de cette phase, les données sont soumises au modèle. Ce modèle applique ses connaissances acquises lors de l'entraînement pour générer des prédictions.

Il est essentiel d'évaluer et d'analyser les prédictions générées par notre modèle en s'appuyant sur des métriques d'évaluation telles que l'exactitude (accuracy), perte de Hamming (Hamming loss) et la F-mesure (F-score) permettent d'évaluer la performance de notre modèle.
Voici l'algorithme de prédiction :

Algorithme 2 : Fonction de prédition

Fonction predict(X)
Entrées :
X : tableau-like ou matrice creuse, shape = [n_samples, n_features]
Les échantillons d'entrée pour lesquels nous souhaitons effectuer des prédictions.
Variables locales :
models : Liste des classifieurs individuels pour chaque classe.
preds : Liste pour stocker les prédictions des classifieurs (initialisée à une liste vide).
Début :
Pour chaque feature_label, affecter sa prédiction
Pour chaque clé dans col_f
X[clé] = 0
# Créer les colonnes des nouvelles features manquantes en leur attribuant des zéros comme valeurs
Pour chaque clé dans col_f
Prédiction = prédiction binaire pour la classe correspondante du modèle : models[dict_labels[clé]] pour X.
Remplacer les valeurs de la colonne correspondante dans X avec les valeurs de Prédiction.
Pour chaque modèle dans models
Prédiction = prédire les étiquettes pour X en utilisant le modèle courant.
Ajouter les prédictions a la liste preds.
Sorties : preds #liste des predictions
Fin de la fonction

2.9 Conclusion

En conclusion, cette partie conception présente notre système proposé pour la classification multi-étiquette en langue arabe en s'inspirant des deux principes de Binary Relevance et Classifier Chains. Notre approche se distingue par l'utilisation de la corrélation entre les étiquettes pour améliorer la précision de la classification.

Chapitre 3

Implémentations et Résultats

3.1 Introduction

Le chapitre précédent a abordé en détail le processus et le cadre conceptuel de notre proposition. Dans le présent chapitre, nous allons présenter le contenu pratique effectué en définissant les méthodes et les outils adoptés pour l'implémentation, ainsi que le prétraitement effectué sur la base de données. Ensuite, nous présentons les tests expérimentaux réalisés en utilisant ces différents modèles, accompagnés d'une discussion approfondie des résultats établies.

3.2 Matériel et outils de développement

3.2.1 Materiel de developpement

Notre système de classification multi-étiquette sera implémenté sur les deux machines dont les caractéristiques sont les suivants :

	Marque	**CPU**	**Ram**	**GPU**	**OS**
PC1	Toshiba	I7-6500U	8 Gb	Intel 540	Windows 10
PC2	MSI	I7 10éme	16 Gb	RTX 2070	Windows 11

TABLE 3.1 – Machines utilisées

3.2.2 Outils de développement

3.2.2.1 Python

Python est un langage de programmation polyvalent et haut niveau, qui se distingue par sa typage dynamique et sa récupération. Il offre une large gamme de bibliothèques et de frameworks dédiés au Machine Learning, tels que Pandas, Numpy, scikit-learn, facilitant le développement et l'implémentation de modèles d'apprentissage automatique [27].

Durant l'étape de pré-traitement plusieurs bibliothèques ont été utilisées :

- **Numpy (Numeric Python) :**
  NumPy est une bibliothèque Python populaire pour le calcul scientifique qui fournit des structures de données avancées et des fonctions performantes pour manipuler et analyser des tableaux multidimensionnels [34].

- **Pandas :**
  Pandas est une bibliothèque Python pour la manipulation et l'analyse de données tabulaires, offrant des structures de données efficaces comme les DataFrames. Elle est largement utilisée dans le domaine de la science des données pour ses fonctionnalités avancées et sa facilité d'utilisation [28].

- **Matplotlib(Mathematic Plot Library) :**
  Matplotlib est une bibliothèque de plotting (traçage) 2D qui produit de nombreux formats matriciels et vectoriels. Elle permet également, une visualisation graphique des données grâce à l'ensemble de représentations graphiques qu'elle propose [29].

- **Scikit-learn :**
  Scikit-learn est une bibliothèque Python populaire pour l'apprentissage automatique. Elle est apprécié pour sa simplicité d'utilisation, sa documentation complète et sa grande communauté de développeurs. Elle est utilisé pour résoudre des problèmes d'apprentissage automatique de manière efficace et fiable [30].

- **Scikit-multilearn :**
  Scikit-multilearn est une bibliothèque Python basée sur scikit-learn qui facilite la manipulation des données multi-étiquettes et propose différents algorithmes de classification, de transformation et d'évaluation adaptés à ce type de tâche. Elle est largement utilisée dans la recherche et les applications pratiques impliquant des problèmes de classification multi-étiquettes [31].

- **NLTK (Natural Language Toolkit) :**
  NLTK est une bibliothèque Python populaire pour le traitement automatique du langage naturel (TALN). Elle offre des fonctionnalités telles que la tokenisation, la lemmatisation, la classification de textes, et la construction de modèles de langage [32].

- **Farasa :**
  Farasa est une bibliothèque de traitement automatique du langage naturel (TALN) spécifiquement conçue pour la langue arabe. Elle propose des fonctionnalités telles

que la tokenisation, la segmentation, la désambiguïsation morphologique et l'analyse syntaxique. Elle offre également des fonctionnalités pour la normalisation des caractères arabes et la résolution des formes fléchies [35].

3.2.3 Plateforme de développement

3.2.3.1 Google colab :

La mémoire vive de notre pc n'est pas suffisante pour certaines tâches, comme Windows 10 a besoin de 4gb pour lui tout seul afin de fonctionner correctement, nous avons préféré exploiter l'outil de google afin d'avoir plus de mémoire pour aboutir aux résultats. Avec les 28 gb de mémoire, nous avons pu avancer mais ça reste assez limité.

3.2.3.2 Jupyter netbook :

Jupyter Notebook est un environnement de développement interactif utilisé pour l'analyse de données, la visualisation et la création de documents interactifs. Il permet d'écrire du code, d'exécuter des cellules de code individuelles, de visualiser les résultats et d'ajouter des textes explicatifs dans un format unique. Les notebooks Jupyter prennent en charge plusieurs langages de programmation, dont Python, R et Julia [33].

3.3 Implémentation de système proposé :

3.3.1 Acquisition des données :

Le dataset BBC est un ensemble de données d'actualité comprenant des articles de presse (en arabe), composé de plusieurs colonnes, dont les plus importantes sont : Article,Main_Class, subclass1, subclass2, subclass3, subclass4, subclass5, nous prenons en exemple la figure 3.1 :

	semi_clean	Clean_Article	Article	Main_class	subclass1	subclass2	subclass3	subclass4	subclass5
0	اتفق سفراء الاتحاد الاوروبي في بروكسل على حزمه...	اتفق سفير اتحاد أوربي بروكسل حزم عقوبة جديد ضد...	اتفق سفراء الاتحاد الأوروبي في بروكسل على حزمة...	أوروبا	روسيا	سياسة	تجارة واعمال	NaN	NaN
1	تراجع الجنيه الاسترليني لادني معدلاته في اشهر ...	تراجع جنيه إسترليني لادن معدل أشهر سبب شك حول ...	تراجع الجنيه الاسترليني لأدني معدلاته في أشهر...	المملكة المتحدة	تجارة واعمال	NaN	NaN	NaN	NaN
2	وقعت شركتا بي بي البريطانيه وسي ان بي سي الصين...	وقع شرك بريطاني سي سي صينيه عقد معدل تضمن زياد...	وقعت شركتا بي بي البريطانية وسي ان بي سي الصين...	تجارة واعمال	العراق	المملكة المتحدة	NaN	NaN	NaN

FIGURE 3.1 – Échantillon de l'ensemble de données BBC.

En analysant notre corpus, nous avons remarqué qu'il souffre d'un déséquilibre entre les différentes classes. En d'autres termes, certaines classes peuvent être largement représentées, tandis que d'autres peuvent être sous-représentées, comme on peut le constater dans la figure 3.2 ci-dessous :

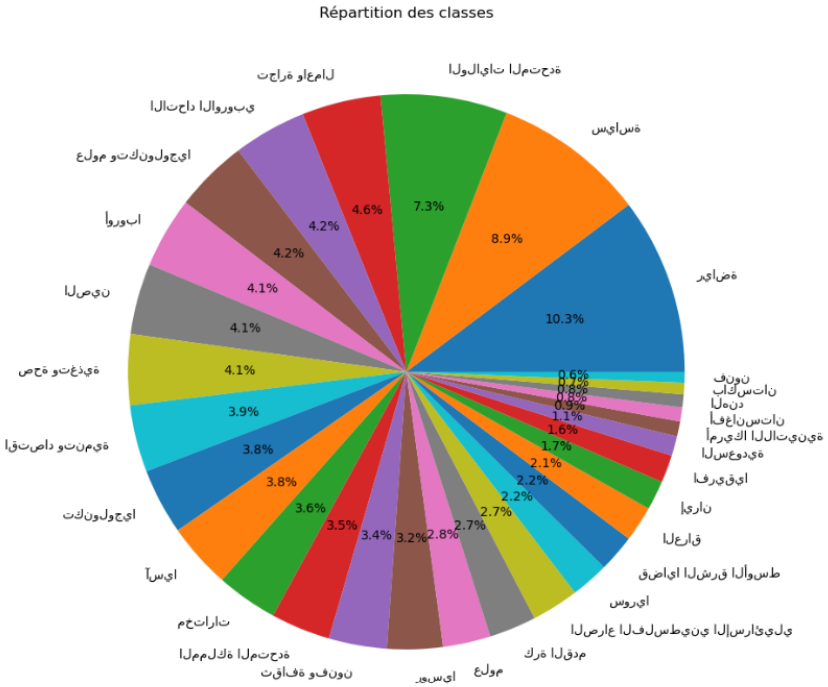

FIGURE 3.2 – Représentation des 30 classes les plus fréquentes.

3.3.2 Processus de prétraitement

La phase du prétraitement des données est une étape cruciale pour garantir la qualité et la pertinence des données textuelles utilisées dans notre étude. Le processus de prétraitement que nous avons réalisé comprend plusieurs étapes.

1. **La suppression des diacritiques :**
   Cette tâche supprime ce qu'on appelle en langue arabe El-Tachkil. Cette tâche a été exécutée en utilisant la bibliothèque regex qui est intégrée dans python, comme le montre la figure 3.3 suivante :

```python
#Supprimer les signes diacritique
def remove_diacritics(string):
    regex = re.compile(r'[\u064B\u064C\u064D\u064E\u064F\u0650\u0651\u0652]')
    return re.sub(regex, '', string)
```

FIGURE 3.3 – Suppression des diacritiques.

2. **La suppression des URL (Uniform Resource Locator) :**
   Ccette tâche supprime toute adresse web ou lien vers un site web, car dans notre approche ce type d'information est jugé non pertinent, comme le montre la figure 3.4

suivante :

```
#supprimer les URLs
def remove_urls(string):
    regex = re.compile(r"(http|https|ftp)://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-f
    return re.sub(regex, ' ', string)
```

FIGURE 3.4 – suppression des URL.

3. **La suppression des espaces supplémentaires :**
   Il arrive parfois de trouver un double espacement en raison d'une faute de frappe, cette tâche supprime tout espace supplémentaire, comme le montre la figure 3.5 suivante :

```
#Supprimer les espaces supplémentaires
def remove_extra_whitespace(string):
    string = re.sub(r'\s+', ' ', string)
    return re.sub(r"\s{2,}", " ", string).strip()
```

FIGURE 3.5 – Suppression des espaces supplémentaires.

4. **La suppression des mots non arabes :**
   Notre étude vise à expérimenter une approche multi-étiquettes sur un texte arabe, de ce fait toute lettre non arabe est considérée comme étant un élément indésirable qui doit être supprimé. Cette tâche consiste à nettoyer notre corpus de tous les mots non arabe, comme le montre la figure 3.6 suivante :

```
#Supprimer les mots non arabes
def remove_no_arabic_words(string):
    return ' '.join([word for word in string.split() if not re.findall(
        r'[^\s\u0621\u0622\u0623\u0624\u0625\u0626\u0627\u0628\u0629\u062A\u062B\u
        word)])
```

FIGURE 3.6 – Suppression des mots non arabes.

5. **La suppression des symboles non arabes :**
   Cette tâche supprime les symboles, comme le montre la figure 3.7 suivante :

```
#Supprimer les symboles non arabes
def remove_no_arabic_symbols(string):
    return re.sub(r'[^\u0600-\u06FF]', ' ', string)
```

FIGURE 3.7 – Suppression des symboles non arabes.

6. **La suppression des nombres :**
   Cette tâche supprime les nombres, comme le montre la figure 3.8 suivante :

```python
#Supprimer les nombres
def remove_numbers(string):
    regex = re.compile(r"(\d|[\u0660\u0661\u0662\u0663\u0664\u0665\u0666\u0667\u0668\u0669])+")
    return re.sub(regex, ' ', string)
```

FIGURE 3.8 – Suppression des nombres.

Après cela nous avons procédé à la normalisation du texte arabe qui consiste à remplacer les ["ة","آ","إ","أ"] par des ["ه","ا","ا","ا"], comme le montre la figure 3.9 :

```python
# Normaliser le Text Arabic
def clean_txt(string):
    search = ["أ","إ","آ","ة"]
    replace = ["ا","ا","ا","ه"]

    string = string.replace('وو', 'و')
    string = string.replace('يي', 'ي')
    string = string.replace('اا', 'ا')

    for i in range(0, len(search)):
        string = string.replace(search[i], replace[i])
    return string
```

FIGURE 3.9 – Normalisation des données.

Subséquemment, la suppression des mots vides est effectuée, comme illustré dans la Figure 3.10 , parmi cet ensemble de mots supprimés on peut citer : [و ، في ، على، من، إلى] (nous tenons à préciser que la liste des mots vides ne se résume pas seulement aux mots cités précédemment).

```python
#Supprimer les stops Words
from nltk.corpus import stopwords
stop_words = set(stopwords.words('arabic'))

mots_vide=[]
for el in stop_words:
  txt=remove_diacritics(el)
  txt=clean_txt(txt)
  mots_vide.append(txt)

def remove_arabic_stopwords(text):
    words = text.split()
    filtered_words = [word for word in words if word not in mots_vide ]
    return ' '.join(filtered_words)
# Appliquer la fonction
clean_arr=[]
for el in data_arr:
  clean_arr.append(remove_arabic_stopwords(el))

clean_arr
```

FIGURE 3.10 – Suppression des mots vides.

Puis à la fin nous avons appliqué la racinisation (stemming) sur notre ensemble de données comme le montre la figure 3.11, dans le but de ramener les mots à leurs formes canoniques en éliminant les suffixes et préfixes. Cela permet de regrouper les variantes d'un même mot et d'améliorer la cohérence des données. On obtient a la fin un jeu de données propre. Après avoir testé plusieurs bibliothèques de prétraitement de la langue arabe comme tashaphyne et NLTK, notre choix s'est porté sur la bibliothèque Farasa pour son rendement qui est supérieur à celui des autres bibliothèques.

La fonction de stemming est représenté dans la figure 3.11 ci-dessous :

```
#Stemming avec Farasa
from farasa.stemmer import FarasaStemmer
stemmer=FarasaStemmer()
arr_stm=[]

def farasa_stem(arr):
    for i in range(len(arr)):
        text=stemmer.stem(arr[i])
        arr_stm.append(text)

farasa_stem(clean_arr)
arr_stm
```

FIGURE 3.11 – Fonction de racinisation.

3.3.3 Extraction des caractéristiques

Nous avons utiliser la méthode TF-IDF (Term Frequency-Inverse Document Frequency) comme le montre la figure 3.12, ce qui nous a permis de représenter les textes en tant que vecteurs numériques en tenant compte de la fréquence des termes dans chaque document et de leur importance globale dans l'ensemble du corpus. Cela nous permet de capturer les caractéristiques distinctives de chaque document et d'obtenir une représentation numérique adaptée à l'apprentissage automatique.

```
from sklearn.feature_extraction.text import TfidfVectorizer

# vectorisation avec TF-iDF
tfidf_vectorizer = TfidfVectorizer(max_features=9000, ngram_range=(1,3))
tfidf_vectors = tfidf_vectorizer.fit_transform(df1.Clean_Article)

df_tfidf = pd.DataFrame(tfidf_vectors.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

# Afficher le DataFrame
df_tfidf.head(10)
```

FIGURE 3.12 – Extraction des caractéristiques avec TF-IDF.

Nous avons configuré TF-IDF avec le paramètre "max-features=9000" afin de sélectionner uniquement les 9000 caractéristiques les plus pertinentes parmi les 25000 existants.

L'ajout du n-gramme à TF-IDF permet d'introduire des informations contextuelles et de sémantique dans la représentation vectorielle des textes. Alors que TF-IDF se concentre principalement sur les occurrences individuelles de termes dans les documents.

Après l'exécution de la fonction de vectorisation du TF-IDF on obtient le résultat suivant qui est illustré dans la figure 3.13 :

	آخر	آخر أضاف	آخر بين	آخر جرح	آخر قال	آخر كان	آخر مبارا	آراء	آلة	آلن	...	يونايتد	يونايتد انجليزي	يونايتد متصدر	يونايتيد
0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	...	0.0	0.0	0.0	0.000000
1	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	...	0.0	0.0	0.0	0.000000
2	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	...	0.0	0.0	0.0	0.000000
3	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.06639	0.0	0.0	...	0.0	0.0	0.0	0.000000
4	0.028084	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	...	0.0	0.0	0.0	0.000000
5	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	...	0.0	0.0	0.0	0.065086
6	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	...	0.0	0.0	0.0	0.000000
7	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	...	0.0	0.0	0.0	0.000000
8	0.023515	0.0	0.0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	...	0.0	0.0	0.0	0.000000

FIGURE 3.13 – Les tokens générés par TF-IDF.

3.3.4 Transformation des labels

Avant d'entamer la classification, les labels doivent passer par un processus de transformation, car au départ a chaque article était attribuer une Mainclass, subclass 1, subclass2, subclass3, subclass4, subclass5, comme colonnes qui contenaient les classes a lesquelles appartient l'article, et pour simplifier la tâche de classification multi-étiquettes on va devoir transformer chaque classe en un problème de classification binaire, et nous devons passer par les étapes suivantes :

- **Fusionner les colonnes :**
  Les colonnes suivantes doivent etre fusionner :Main_class, subclass1, subclass2, subclass, subclass4, subclass5, en une seule colonne (Classes) contenant toutes les classes a lesquelles appartient l'article, la figure 3.14 nous montre plus en détail cette fusion :

```
# transformer les labels en une seule label
df['classes']=df.apply(lambda row: ','.join([str(row[col]) for col in labels if str(row[col]) !='

df[['classes']].head()
```

Python

	classes
0	أوروبا,روسيا,سياسة,تجارة واعمال
1	المملكة المتحدة,تجارة واعمال
2	تجارة واعمال,العراق,المملكة المتحدة
3	تجارة واعمال,المملكة المتحدة
4	تجارة واعمال,المملكة المتحدة,تكنولوجيا

FIGURE 3.14 – Création de la colonne Classes.

- Mettre les classes dans une liste :
  Les classes doivent etre regrouper dans une liste pour pouvoir faire le One Hot Encoding par la suite, la figure 3.15 illustre cette étape :

FIGURE 3.15 – Rassembler les classes dans une liste.

- La figure 3.16 montre comment on a procédé au one hot encoding pour les labels, en utilisant MultiLabelBinarizer de scikitlearn pour transformer une liste de classes multi-label en une représentation binaire.

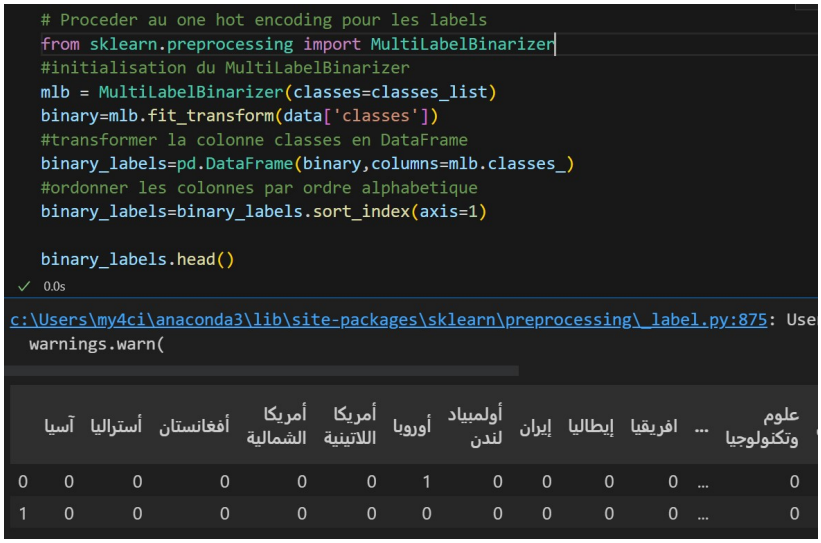

FIGURE 3.16 – One hot encoding.

3.3.5 Classification

Comme mentionné déjà dans la section **"Classification multi-étiquette (2.8)"** du chapitre 02, le système de classification multi étiquettes de langue arabe qu'on propose suit 4 grandes parties.

- **Calculer la matrice de corrélation :**
  La figure ci-dessous montre la fonction que nous avons utilisée pour calculer la matrice de coefficients de corrélation. Dans notre étude nous avons choisi le coefficient de corrélation de Pearson :

```python
# verifier si la matrice correlation existe
if self.corr_mat is None:
    # claculer la matrice de coef de correlation de Pearson
    self.corr_mat = y.corr()

# remplacer le val de la diagonale par 0
np.fill_diagonal(self.corr_mat.values, 0)
# verifier si on a seuil en params
if self.seuil is None:
    # attribuer la moyenne de la matrice de corr comme seuil
    self.seuil = self.corr_mat.mean().mean()
```

FIGURE 3.17 – Calcule de la matrice de corrélation.

La figure 3.18 montre une partie de la matrice obtenue :

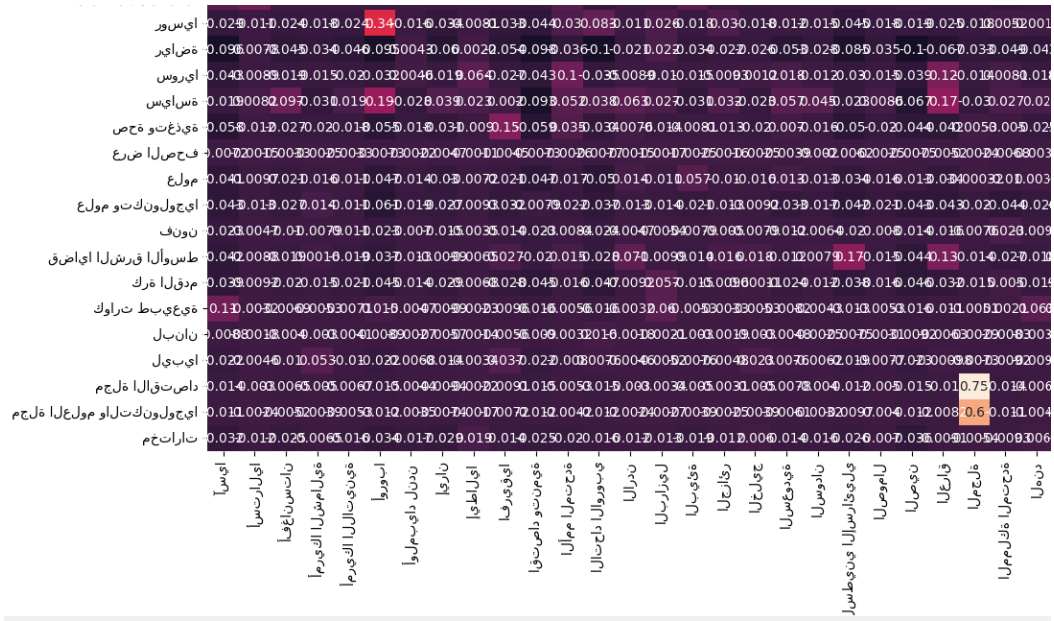

FIGURE 3.18 – Matrice de corrélation de pearson.

Cette matrice serra par la suite donnée à notre algorithme de classification BRC en paramètre pour l'utiliser à l'extraction des étiquettes corrélées et pour calculer le seuil. Si on attribue pas un seuil comme paramètre à notre classifieur alors la moyenne de la matrice de corrélation sera considérée comme seuil.

- **Génération des listes d'étiquettes corrélées :**
  En analysant notre matrice, nous observons les valeurs de corrélation entre les étiquettes ce qui nous a permis d'identifier un seuil qui représente une corrélation significative. Ce seuil, fixé a 0.5, sera utilisé pour générer la liste des étiquettes corrélées.

```
arr=cor_mat[['المجلة']].loc[cor_mat['المجلة'].apply(lambda x: x > 0.5 )].T.columns
list(arr)
✓ 0.0s
['مجلة الاقتصاد', 'مجلة العلوم والتكنولوجيا']
```

FIGURE 3.19 – Génération des listes d'étiquettes corrélées.

- **Entrainer notre modele :**
  Par la suite, la matrice de corrélation et le seuil ainsi qu'un classifieur qu'on choisit, par exemple NB, seront prit comme paramètre dans le but de prendre en compte les dépendances entre les étiquettes. Les algorithmes que nous avons testé sont : NB, SVM, DT, RF, leurs résultats sont résumés dans un tableau dans la section **"Résultat (3.5)"** un peu plus loin dans ce chapitre.
  La figure 3.20 montre comment on a généré les nouvelles liste des caractéristiques.

```
i=0
# pour chaque classe
for label in list(y.columns):
    # remplir notre dict pour stocker key:label et value:indice
    self.dict_labels[label]=i
    i+=1
    # voir si la classe est corrélé avec d'autres classe ou pas
    corr_class=self.corr_mat[[label]].loc[self.corr_mat[label].apply(lambda x: x > self.seu
    if len(corr_class) > 0:
        for cl in list(corr_class):
            X_copy[cl]=y.loc[:,cl]
```

FIGURE 3.20 – Nouvelle liste des caracteristiques.

Pour par la suite entraîner notre modèle avec les nouvelles caractéristiques pour chaque étiquette comme le montre la figure ci dessous :

```
# extraire les colonne des nouveaux Features
self.col_f=X_copy.iloc[:,taille:].columns.tolist()

# pour chaque classe
for label in list(y.columns):
    # Vérifier que X et y ont la bonne forme
    x_checked, y_checked = check_X_y(X_copy, y[label])
    # chaque classifieur est indépendant des autres
    # donc nous créons une copie de l'instance du classifieur de base
    base_model = clone(self.base_classifier)
    # ajuster le modèle de base - un modèle pour chaque Y1, Y2....Y14
    base_model.fit(x_checked, y_checked)
    # ajouter le modèle ajusté à la liste des classifieurs individuels
    self.models.append(base_model)
```

FIGURE 3.21 – Entraînement avec les nouvelles caractéristiques.

- **Faire la prédiction :**
  Avant d'entamer la prédiction nous devons d'abord ajoutés les colonnes des nouvelles caractéristiques sur lesquelles notre modèle s'est entraîné en les initialisant à 0 au départ pour ne pas tomber sur le problème de dimensionnalité, apres cela prédiction sera faite seulement sur la liste des étiquettes corrélées et leurs prédiction seront affectées au nouvelles caractéristiques, la figure 3.22 illustre ce qu'on vient d'expliquer :

```python
def predict(self, X):
    # vérifier si la liste des modèles a été configurée
    check_is_fitted(self, ['models'])
    # crée une copie de X
    X_copy=X.copy()
    # crée les colonnes des features manquant en leurs attribuant 0 comme val
    for cl in self.col_f:
        X_copy[cl]=0

    Xpred=X_copy.copy()

    # pour chaque feature_label affecté sa prédiction
    for key in self.col_f:
        Xpred[key]=self.models[self.dict_labels[key]].predict(X_copy).tolist()
```

FIGURE 3.22 – Ajouter les nouvelles caractéristiques pour la prédiction

Après l'obtention des nouveaux caractéristique nous procédons alors à la prédiction de chaque étiquette avec les nouveaux caractéristique individuellement comme le montre la figure ci dessous :

```python
all_preds = pd.DataFrame()
i=0
# liste des prédictions des classifieurs individuels
preds = []

# prédire pour chaque modèle ajusté - un modèle par étiquette
for model in self.models:
    pred = model.predict(Xpred)
    # ajouter la prédiction au dataframe
    preds.append(pd.DataFrame({'Classe'+ str(i+1): pred}))
    i+=1

# dataframe avec les prédictions pour toutes les étiquettes de
all_preds = pd.concat(preds, axis=1)
# les classifieurs standard de sklearn retournent les prédictio
# donc convertir le dataframe en un tableau numpy
return all_preds.to_numpy()
```

FIGURE 3.23 – Prédiction.

3.4 Résultats et discussion

Avant de tester notre aproche une classification mono-étiquettes a été effectué sur notre jeu de données BBC, nous avons par la suite comparé nos résultats mono-étiquettes avec les résultats de Binary Relevance (sans corrélation), dans la section qui suite, section "**Comparaison mono-étiquettes 3.5.2"**.

Pour évaluer les résultats de notre algorithmes BRC, on ne s'est pas arreter seulement sur les mesure d'évaluation telle que la perte de hamming, l'exactitude... etc, nous l'avons aussi

comparé au BR normal (sans corrélation) dans le but de voir l'importance des dépendances intra-étiquettes dans classification multi-étiquettes, cette comparaison est abordée plus en détail dans la section **"comparaison BRC 3.5.3"**.

3.4.1 Mesures d'évaluation :

Les mesures d'évaluation jouent un rôle essentiel dans l'analyse des performances des modèles de classification dans le domaine du traitement automatique du langage naturel (TALN). Nous passons en revue certaines mesures de corrélation couramment utilisées pour évaluer les performances des modèles de classification multi-label, à savoir l'accuracy, la hamming loss et le F1-score :

L'accuracy (ou l'exactitude), est une mesure de corrélation classique utilisée pour évaluer la performance globale d'un modèle de classification multi-label. Elle représente le rapport entre le nombre d'instances correctement classées et le nombre total d'instances.Pour la calculer, on divise la somme des vrais positifs et des vrais négatifs par le nombre total d'instances [40].

Hamming loss, est une mesure pertinente pour la classification multi-label, quantifie le degré de désaccord moyen entre les prédictions du modèle et les étiquettes réelles. Elle mesure la fraction moyenne de positions où les prédictions diffèrent des étiquettes réelles. Une hamming loss plus faible indique une meilleure performance du modèle. Pour la calculer, on divise le nombre de positions où les prédictions diffèrent des étiquettes réelles par le nombre total de positions [40].

Le F1-score, est une mesure qui combine la précision et le rappel pour une classe spécifique. Il est couramment utilisé dans la classification multi-label pour évaluer la performance de chaque classe individuellement. Le F1-score est calculé en prenant la moyenne harmonique de la précision et du rappel. La précision mesure la proportion de vrais positifs parmi les instances prédites comme positives, tandis que le rappel évalue la proportion de vrais positifs parmi les instances réellement positives [40].

Il existe d'autres mesures d'évaluation, mais nous avons choisi l'accuracy, la hamming loss et le F1-score car elles fournissent des indicateurs essentiels pour évaluer les performances des modèles de classification multi-label dans le domaine du TALN.

3.4.2 Comparaison de l'algorithme BRC :

Le tableau 3.2 ci-dessous montre plus en détails la comparaison entre les résultats de notre systeme de classification multi-étiquettes (BRC) et les résultats du BR normal :

Résultat	**BR(normal)**				**BRC**			
	NB	**SVM**	**DT**	**RF**	**NB**	**SVM**	**DT**	**RF**
A	36.5	44.5	45.5	50.3	35.8	44.4	43.4	50.8
H	1.7	1.4	1.8	1.3	1.7	1.4	1.9	1.3
F1	55.5	66.7	65.6	68	54.7	66.3	64.6	68.2
T	97	1399	903	682	62	1004	250	266

TABLE 3.2 – Table des résultats de BR et BRC.

D'après les résultats du tableau ci-dessus, on remarque que en utilisant l'algorithme BRC, le modèle de Random Forest a obtenu une performance plus elevée comparé au BR normal. En revanche, les modèles de Naive Bayes, SVM et Decision Tree ont obtenu des performances inférieures.

La figure 3.26 illustre de façon plus claire la comparaison entre les accuracies du BR et BRC :

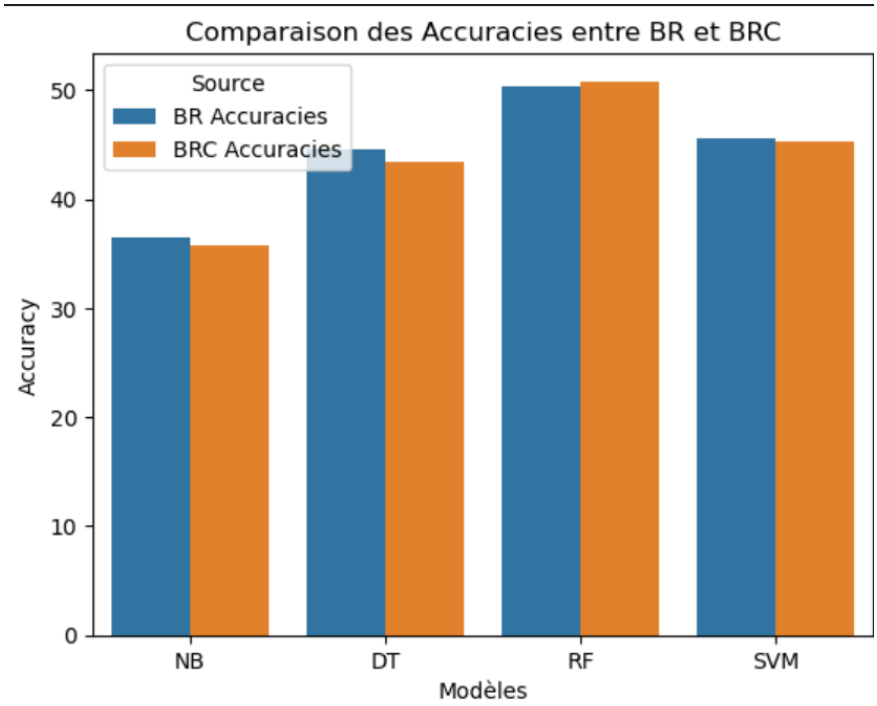

FIGURE 3.24 – Comparaison entre l'accuracy de BR et BRC.

Parmi les 4 modèles utilisés pour tester notre systeme de classification multi-étiquettes a savoir NB, SVM, DT et RF, on remarque une augmentation de 0.5% pour le BRC en utilisant le modèle RF :

-Pour le BR normal on a une exactitude (accuracy) égale à : 50.3

-Pour le BRC on a une exactitude égale à : 50.8

Nous tenons aussi à attirer l'attention sur le temps d'exécution. En effet, nous avons contribué à réduire le temps d'exécution de façon considérable comme on peut le voir sur le tableau 3.2 déjà mentionné ci dessus. Par exemple dans le BR normal le temps d'exécution pour les deux algorithmes SVM et DT est respectivement égale à 1399 et 903, tandis qu'avec notre approche nous avons obtenu 1004 et 250 respectivement avec SVM et DT.

Notre approche a apporté une contribution (même si l'amélioration est assez faible), ce qui montre l'importance de prendre en compte la corrélation dans classification multi-étiquettes. Ces résultats peuvent être améliorés en utilisant d'autres matrices ou mesures de corrélation, ou bien en utilisant un jeu de données plus riche et plus équilibré.

3.5 Difficultés rencontrées

Comparée à la classification multi-étiquettes en langue anglaise, la classification multi-étiquettes en langue arabe est en retard. En effet, peu de recherches ont été menées sur la classification des textes arabes sur des petits ensembles de données dont la majorité est non accessibles au public et plusieurs chercheurs se plaignent eux aussi du même problème, à savoir le manque de corpus.[19], [2].

Mais aussi la complexité de la langue arabe, au niveau grammatical comme au niveau syntaxique la rend difficile à traiter et nécéssite une adaptatiion du processus de prétraitement [2].

Il est important de souligner que la taille et la nature des données utilisées pour l'entraînement des modèles jouent un role important. Un jeu de données plus riche et représentatif pourrait permettre d'améliorer les performances de notre approche. De plus, l'équilibrage des classes pourrait être essentiel pour certaines approches de classification multi-label, car certaines classes sont sous-représentées dans notre corpus, ce qui a affecter nos résultats, comme on peut le voir dans la figure 3.2 de la section **"acquisition des données 3.3.1"**.

3.6 Conclusion

En conclusion de ce dernier chapitre d'implémentation, nous avons examiné les outils de développement utilisés. Nous avons décrit le processus de prétraitement des données. Nous avons également extrait les caractéristiques pertinentes des données textuelles. Nous avons appliqué la transformation des labels en utilisant la méthode de la transformation binaire. Enfin, nous avons utilisé notre approche BRC pour la classification multilabel. Les difficultés rencontrées ont été citées. Les résultats obtenus ont démontré l'efficacité de notre approche, avec une augmentation de la précision par rapport au BR normal. Ces résultats prometteurs valident l'importance de la corrélation entre les étiquettes dans la classification du texte et

soulignent le potentiel de notre méthode pour améliorer la précision des classifications multi-étiquettes.

Conclusion générale et perspectives :

En conclusion, notre mémoire de fin d'étude a exploré le domaine de la classification multi-label appliquée à la langue arabe dans le contexte du traitement automatique du langage naturel. Notre objectif était de proposer une approche novatrice basée sur l'exploitation de la corrélation entre les différentes classes du jeu de données.

Dans notre approche proposée, nous avons mis l'accent sur l'exploitation de la corrélation entre les classes du jeu de données. En considérant ces relations, nous avons cherché à mieux capturer la diversité et la richesse des textes en arabe, afin d'améliorer la précision et la performance de la classification multi-label.

La conception de notre approche a été le fruit d'une réflexion minutieuse et d'une recherche bibliographique approfondie. Nous avons examiné les travaux existants dans le domaine de ça classification multi-label en arabe. La recherche de solutions innovantes pour exploiter la corrélation entre les différentes classes a été un défi intellectuel stimulant, nécessitant une créativité et une ingéniosité constantes.

L'implémentation de notre approche a été une étape cruciale pour mettre nos idées à l'épreuve. Les aspects techniques ont été exigeants, notamment en raison des spécificités de la langue arabe, telles que les variations dialectales etc. Cependant, nous avons surmonté ces difficultés en exploitant les ressources et les outils disponibles, et en adaptant notre approche aux particularités de la langue arabe.

Meme si les résultats obtenu grace a notre algorithme BRC ne sont pas a la hauteur de nos attentes, nous sommes parvenus à apporter une contribution à la classification multi-étiquettes en la langue arabe, ce qui confirme l'importance de prendre en compte la corrélation entre les différentes catégories du jeu de données.

Cependant, notre travail comporte certaines limites. Tout d'abord, le manque des corpus multi-étiquettes en langue arabe, mais aussi le problème de déséquilibre de données dont souffre notre jeu de données. De plus, notre approche pourrait bénéficier d'une exploration plus approfondie des différentes techniques d'exploitation de la corrélation entre les classes.

Malgré ces limites, notre mémoire de fin d'étude représente une contribution au domaine du traitement automatique du langage arabe, en proposant une nouvelle approche basée sur l'exploitation de la corrélation entre les différentes classes du jeu de données. Cette approche ouvre de nouvelles perspectives pour la classification multi-label en arabe et offre des opportunités d'amélioration dans la compréhension et l'analyse des textes en langue

arabe.

Pour les travaux futurs, il serait intéressant d'explorer davantage les méthodes d'exploitation de la corrélation entre les classes, en utilisant des techniques telles que l'apprentissage profond et l'analyse sémantique avancée. De plus, l'extension de nos expérimentations à des jeux de données plus vastes permettrait de renforcer les résultats obtenus.

En conclusion, notre recherche offre une contribution à l'avancement du traitement automatique du langage arabe, en proposant une approche novatrice pour la classification multi-label basée sur l'exploitation de la corrélation entre les différentes classes. Nous espérons que ce mémoire constituera une référence précieuse pour les chercheurs et les praticiens intéressés par l'amélioration des performances de classification en arabe, et nous encourageons la poursuite des travaux dans ce domaine passionnant et en constante évolution.

Bibliographie

[1] Matallah, H.(2011). Classification automatique de textes : approche orientée agent. 6–21.

[2] Aljedani, N., Alotaibi, R., & Taileb, M. (2020). Multi-label arabic text classification : an overview. International Journal of Advanced Computer Science and Applications, 11(10), 694–706. https://doi.org/10.14569/IJACSA.2020.0111086

[3] Bogatinovski, J., Todorovski, L., Džeroski, S., & Kocev, D. (2022). Comprehensive comparative study of multi-label classification methods. Expert Systems with Applications, 203. https://doi.org/10.1016/j.eswa.2022.117215

[4] Doumi, N., Lehireche, A., Maurel, D., & Ali, M. (2014). Conception d ' un jeu de ressources libres pour le TAL arabe sous Unitex. 1–15.

[5] Fahed, L., Frey, G., Thompson, J., Lachiche, N., & Thompson, J. D. (2013). Classification multi-étiquettes pour l'alignement multiple de séquences protéiques Rare event prediction / Industry 4.0 View project Semantic technologies for the optimization of complex biomolecular networks View project Classification multi-étiquettes pour l'alignement multiple de séquences protéiques. https://www.researchgate.net/publication/273384154

[6] Guellil, I., Saâdane, H., Azouaou, F., Gueni, B., & Nouvel, D. (2021). Arabic natural language processing : An overview. In Journal of King Saud University - Computer and Information Sciences (Vol. 33, Issue 5, pp. 497–507). King Saud bin Abdulaziz University. https://doi.org/10.1016/j.jksuci.2019.02.006

[7] Khorsheed, M. S., & Al-Thubaity, A. O. (2013). Comparative evaluation of text classification techniques using a large diverse Arabic dataset. Language Resources and Evaluation, 47(2), 513–538. https://doi.org/10.1007/s10579-013-9221-8

[8] Uysal, A. K. (2016). An improved global feature selection scheme for text classification. Expert Systems with Applications, 43, 82–92. https://doi.org/10.1016/j.eswa.2015.08.050

[9] Debili, Fathi & Hadhémi, Achour & Souissi, Emna. (2002). La langue arabe et l'ordinateur de l'étiquetage grammatical à la voyellation automatique. Correspondances : bulletin de l'IRMC, ISSN 0330-7417, Nº 71, 2002, p. 10-26.

[10] Abdennour, B. (2020). Vers une plateforme de gestion des corpus et d'analyse de texte en langue arabe .

[11] Al-Saleem, S. (2010). Associative classification to categorize Arabic data sets. The International Journal Of ACM JORDAN, 1, 118–127.

[12] Ayyoub, A. (2015). classification de texte arabe multi étiquettes évolutive.

[13] Luaces, O., Díez, J., Barranquero, J., del Coz, J. J., & Bahamonde, A. (2012). Binary relevance efficacy for multilabel classification. Progress in Artificial Intelligence, 1(4), 303–313. https://doi.org/10.1007/s13748-012-0030-x

[14] Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. Machine Learning, 85(3), 333–359. https://doi.org/10.1007/s10994-011-5256-5

[15] Shubham, J., 2017, Solving Multi-Label Classification problems, analyticsvidhya, https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/, (Consulté le 02 mai 2023)

[16] Read, J., Pfahringer, B., & Holmes, G. (2008). Multi-label classification using ensembles of pruned sets. Proceedings - IEEE International Conference on Data Mining, ICDM, 995–1000. https://doi.org/10.1109/ICDM.2008.74

[17] Brownlee, J., (2018), How to Calculate Correlation Between Variables in Python, https://machinelearningmastery.com/how-to-use-correlation-to-understand-the-relationship-between-variables/

[18] Pritha, B., (2021), Correlation Coefficient Types, Formulas & Examples, https://www.scribbr.coom/statistics/correlation-coefficient/

[19] Al-salemi, B., Ayob, M., Kendall, G., Azman, S., & Noah, M. (2019). Multi-label Arabic text categorization : A benchmark and baseline comparison of multi-label learning algorithms. Information Processing and Management, 56(1), 212–227. https://doi.org/10.1016/j.ipm.2018.09.008

[20] Taha, A. Y., & Tiun, S. (2016). Binary relevance (BR) method classifier of multi-label classification for arabic text. Journal of Theoretical and Applied Information Technology, 84(3), 414–422.

[21] Ahmed, N. A., Shehab, M. A., Al-ayyoub, M., & Hmeidi, I. (2015). Scalable Multi-Label Arabic Text Classification. 212–217.

[22] Shehab, M. A., Badarneh, O., Al-Ayyoub, M., & Jararweh, Y. (2016). A supervised approach for multi-label classification of Arabic news articles. Proceedings - CSIT 2016 : 2016 7th International Conference on Computer Science and Information Technology, 1–6. https://doi.org/10.1109/CSIT.2016.7549465

[23] Tsoumakas, G., Katakis, I., & Vlahavas, I. (2010). "Random k-labelsets : An ensemble method for multilabel classification," in European conference on machine learning. Springer, 2007, pp. 406–417.

[24] Smith, J., & Johnson, A. (2021). Understanding TF-IDF : A comprehensive analysis. Journal of Information Retrieval, 15(2), 45-68.

[25] De Roeck, A., N., Nanas, N. (2004). Comparative analysis of weighting methods for information retrieval. Journal of Information Science, 26(4), 789-812.

[26] Cheikh, M. (2004). Prétraitement de texte arabe en python, à partir de https://www.moustaphacheikh.com/posts/arabic-text-preprocessing/

[27] Python. (Sans date). Python Programming Language - Official Website. Récupéré le 23 mai 2023, à partir de `https://www.python.org`

[28] Pandas. (sans date). Pandas - Python Data Analysis Library. Récupéré le 23 mai 2023, à partir de `https://pandas.pydata.org`

[29] Matplotlib. (sans date). Matplotlib : Visualization with Python. Récupéré le 23 mai 2023, à partir de `https://matplotlib.org`

[30] Scikit-learn. (sans date). Scikit-learn : Machine Learning in Python. Récupéré le 23 mai 2023, à partir de `https://scikit-learn.org/`

[31] Scikit-multilearn. (sans date). Scikit-multilearn : A scikit-based Python environment for performing multi-label classification. Récupéré le 23 mai 2023, à partir de `http://scikit.ml/`

[32] NLTK. (sans date). Natural Language Toolkit - NLTK 3.6 documentation. Récupéré le 23 mai 2023, à partir de `https://www.nltk.org/`

[33] Project Jupyter. (sans date). Jupyter Notebook. Récupéré le 23 mai 2023, à partir de `https://jupyter.org`

[34] NumPy. (sans date). NumPy : The fundamental package for scientific computing with Python. Récupéré le 23 mai 2023, à partir de `https://numpy.org`

[35] Farasa. (sans date). Farasa - Arabic Natural Language Processing Toolkit. Récupéré le 23 mai 2023, à partir de `https://farasa.qcri.org/`

[36] Benhamida, A. (2020). Vers une plateforme de gestion des corpus et d 'analyse de texte en langue arabe .

[37] Nizar, Shehab, Mahmoud, Al-ayyoub, & Hmeidi, 2015). Scalable Multi-Label Arabic Text Classification. 212–217

[38] Khorsheed, M. S., & Al-Thubaity, A. O. (2013). Comparative evaluation of text classification techniques using a large diverse Arabic dataset. Language Resources and Evaluation, 47(2), 513–538. `https://doi.org/10.1007/s10579-013-9221-8`

[39] Gherabi, S. (2013). CLASSIFICATION AUTOMATIQUE DES TEXTES ARABE

[40] . Tsoumakas, I. Katakis et I. Vlahavas, « Mining multi-label data», dans Data mining and knowledge discovery handbook. Springer, 2010.