Université Saad DAHLAB - Blida 1

Faculté des sciences

Département d'Informatique

Mémoire présenté par :

Mlles. BENALIA Soumia et OTMANI Marwa

Pour l'obtention du diplôme de Master

Domaine : Mathématique et Informatique

Filière : Informatique

Spécialité : Traitement Automatique de la Langue

Sujet :

Identification des Langues proches et langues lointaines (Latin versus Arabe)

Soutenu le : 08/10/2020, devant le jury composé de

Mr. HAMOUDA	- Université de Blida 1	Présidente
Mme. CHERIGUENE	- Université de Blida 1	Examinatrice
Mr. M.ABBAS	- CRSTDLA	Encadreur
Mme. M. BENBLIDIA	-Université de Blida 1	Promotrice

Résumé

Notre projet de fin d'étude traite l'Identification automatique des Langues qui est au centre de tous les intérêts de ces dernières années. En effet, ce problème a trouvé des intérêts dans diverses sous-disciplines du Traitement Automatique de la Langue, comme la Traduction Automatique, la Correction Orthographique, la Recherche d'Information etc.

Nous avons concentré sur les principaux défis comme la taille de fragment et la vitesse d'exécution et les langues proches. Pour cela, nous avons utilisé un corpus des données étiquetées qui met ensemble six langues (Français, Anglais, Allemand, Arabe, Ourdou, et Persan). Le choix de ces langues a été motivé par la rareté des travaux qui se sont intéressés à ces langues-ci dans le domaine de Détection de Langue.

Ce travail adopte les principales méthodes de classification utilisées en Machine Learning pour s'attaquer à notre problématique. En l'occurrence, nous sommes intéressées aux méthodes : Bayésienne, Machine à Vecteurs de Support, Régression Logistique, Forêts aléatoires, Stimulation des gradients, Algorithme des plus Proches Voisins, Arbres de Décision pour but de trouver la meilleure méthode pour identifier une langue. Nous avons aussi testés des approches linguistiques basées sur les lettres et n-grammes des lettres.

Notre solution a atteint une précision de 99.98 % pour une phrase et de 81% pour un mot de 3 caractères.

Mots clés : Identification de la langue, Classification supervisée, Catégorisation de texte, Reconnaissance de la langue, machine Learning, N-gramme, identification linguistique.

الملخص:

مشروع التخرج الخاص بنا يعمل على تحديد اللغة التلقائي الذي هو في مركز جميع الاهتمامات في السنوات الأخيرة. في الواقع، تجد هذه المشكلة اهتمامًا في العديد من التخصصات الفرعية للمعالجة التلقائية للغة، مثل الترجمة الآلية، والتصحيح الإملائي، واسترجاع المعلومات، إلخ.

ركزنا على التحديات الرئيسية مثل حجم الجزء وسرعة التنفيذ واللغات القريبة. لهذا، استخدمنا مجموعة من البيانات المصنفة التي جمعت ست لغات (الفرنسية والإنجليزية والألمانية والعربية والأردية والفارسية). الدافع وراء اختيار هذه اللغات هو ندرة الأعمال التي اهتمت بهذه اللغات في مجال تحديد اللغة.

يعتمد هذا العمل على طرق التصنيف الرئيسية المستخدمة في التعلم الآلي لمعالجة مشكلتنا. في هذه الحالة، نحن مهتمون بالطرق التالية: بابيزيان، آلة المتجهات الداعمة، الانحدار اللوجستي، الغابات العشوائية، تحفيز التدرجات، خوارزمية الجوار الأقرب، أشجار القرار للعثور على أفضل طريقة لتحديد اللغة. اختبرنا أيضًا الأساليب اللغوية بناءً على الحروف و n-gramsمن الحروف.

لقد استطاع حلنا تحقيق دقة 99.98٪ بالنسبة للجملة و81٪ بالنسبة للكلمة المكونة من 3 أحرف.

<u>الكلمات المفتاحية:</u> تحديد اللغة، تصنيف النصوص، التعرف على اللغة، التعلم الالي، التحديد اللغوي.

Summary :

Our end of study project deals with Automatic Language Identification, which has been the focus of interest recent years. In fact, this problem has found interest in various sub-disciplines of Automatic Language Processing, such as Automatic Translation, Spelling Correction, Information Retrieval, etc.

We focused on the main challenges such as fragment size and execution speed and close languages. For this, we used a corpus of label data that brings together six different languages (French, English, German, Arabic, Urdu, and Persian). The choice of these languages was motivated by the scarcity of work that has focused on these languages in the field of Language Detection.

This work adopts the main classification methods used in Machine Learning to tackle our problem. In this case, we are interested in the methods: Bayesian, Support Vector Machine, Logistic Regression, Random Forests, Gradient Stimulation, Nearest Neighbor Algorithm, Decision Trees in order to find the best method to identify a language. We have also tested linguistic approaches based on letters and n-grams of letters.

Our solution reached a precision of 99.98% for a sentence and 81% for a 3-character word.

Keywords: Language Identification, Supervised Classification, Text Categorization, Language Recognition, Machine Learning, N-gram, Linguistic Identification.

Remerciements

Nous remercions Dieu le tout puissant de nous avoir données la santé, la volonté, le courage d'achever ce projet de fin d'étude.

Ces quelques lignes ne suffirent pas à exprimer toute la gratitude et tous les moments que nous avons partagés ensemble, mais parfois un simple merci vaut bien plus que tous les discours du monde, ce mémoire nous a permises de rencontrer également des personnes merveilleuses et d'une rare générosité.

Nous remercions notre promotrice Professeur Mme Ben Belidia pour son acceptation de diriger notre travail et ses conseils et remarques constructives.

Sans oublier Professeur Mme Mezzi que nous remercions énergiquement pour son soutien et sa générosité de nous aider à finaliser notre mémoire jusque-là, Pour bien vouloir diriger ce travail et pour ses conseils et ses remarques constructives. Nous la remercions particulièrement pour sa disponibilité, ses encouragements. Que dieu la protège cette Ange.

Nous remercions Mr ABBAS, notre encadreur au sein de Centre de Recherche Scientifique et Technique pour le Développement de la Langue Arabe (CRSTDLA), pour nous avoir accordées sa confiance pour la réalisation de ce projet à distance.

Nous remercions Mr LICHOURI, pour ses précieuses directives qu'il nous a prodiguées avec tout intérêt, ainsi que son appuie considérable dans notre démarche, et surtout son disponibilité a tout moment.

Nous adressons nos vifs remerciements aux membres du jury d'avoir bien accepté de consacrer leur temps afin d'évaluer notre projet.

Dédicaces

Je dédie ce modeste travail qui n'aurai jamais pu voir les jours sans les soutiens indéfectibles et sans limite de mes parents qui ne cessent de me donner avec amour nécessaire pour que je puisse arriver à ce que je suis aujourd'hui que dieu vous protège et que la réussite soit toujours à ma portée pour que je puisse vous combler de bonheur.

Je dédie ce travail à :

Mes chers frères Lakder, Yasser, Taha, Yousef.

Je dédie aussi ce travail à tous mes amies et mon binôme.

MARWA

Dédicaces

A ma famille à ma mère, mon père, mes sœurs, mon frère pour votre patience et votre soutien tout au long de ces longues études. Soyez vivement remerciés pour tout le soutien.
Que vous m'avez apportée. Les mots ne suffisent pas à m'exprimer *que Dieu vous protège pour moi, je vous aime et un grand merci.*

A mes amies et toute personne que j'ai rencontrée durant mes années d'études Merci pour tous ces moments de joie et de bonne humeur durant nos études.

SOUMIA

Table des matières

Liste des figures

Liste de tableaux

Liste des abréviations

LID : Identification Automatique de la Langue

TAL : Traitement Automatique de la Langue

TF : Fréquence de Terme

IDF : Fréquence d'Inverse de Document

TF_IDF : Terme Fréquence – Fréquence Inverse du Document

SVM : Machine Vecteur de Support

KNN : Plus Proche Voisin

CT : Catégorisation de Texte

ML : Machine Learning

IA : Intelligence Artificielle

AD : Arbre de Décision

FA : Foret Aléatoire

BNB : Bernoulli Naïve Bayes

MNB : Multinomiale Naïve Bayes

GBM : Gradient Boosting Machine

PrREL : la Liste d'exception des Règles Préfixes

PrGEL : la Liste d' la Liste d'exception globale de Postfix

ACL : la Liste d'Ajout de Caractères

LWS : Alight Weight Stemmer pour la langue Ourdou

BOW : Bag-Of-Word

SWL : la Liste des Mot Vides

GpEL : la Liste d'exception Globale de Prefixe

GsEL : la Liste d’exception Globale de Suffixe

NB : Naïve Bayes

MNG : l’Approche des N-grammes Modifies

HMM : Modèle de Markov Cachés

Big5 : Codage des Cinq Grands

Shift-JIS : Codage de Caractère pour la Langue japonaise

GuoBiao : Codage pour les Caractères

UTF-8 : Universal Character Set

RI : Recherche Information

RL : Régression Logistique

LM : Modélisation du Langue

CGC : Centroide Géométrique Cumulé

CFC : Class Feature Centroid

ICF : Fréquence de classe inverse

CCA : Arithmetic Average Centroid

Introduction général

INTRODUCTION GENERAL

Ces dernières années sont marquées par une augmentation énorme de la quantité d'information électronique rédigée en plusieurs langues. Dans le Traitement Automatique de la langue naturelle ceci devient une tâche nécessaire et difficile à la fois. La détection de la langue est considérée comme une première étape importante pour toute une série de tâches de traitement des langues.

Du fait que la détection de la langue intervient, de nos jours, dans diverses applications dans de nombreux domaines, on résulte qu'elle est devenue un sujet de recherche important. De ce fait, l'objectif de ce travail est de trouver une méthode automatisée capable d'analyser des documents pour identifier la langue dans laquelle ils sont écris.

Le problème de la détermination de la langue d'un passage textuel écrit parmi un ensemble de langues potentielles est un problème complexe vu qu'en effet, un texte peut être écrit en plus d'une langue, mais pour des raisons de simplification, nous faisons ici l'hypothèse typique qu'un texte ne peut être écrit qu'en une seule langue et donc le système de détection de langue ne doit retourner qu'une seule langue correcte à la fin.

Et comme la plupart des recherches se concentrent sur les langues les plus parlées, alors que les langues peu dotées en ressources sont ignorées. Nous avons choisi une base de données variées entre des langues déjà traitées comme anglais et des langues en cours de traitement et difficile comme le Persan et l'Ourdou à cause du degré de richesse morphologique qui est plus élevé dans ces langues.

La détection de la langue est très sensible à la longueur du texte et limitée par la quantité d'information disponible. Certains chercheurs supposent qu'un texte de longueur de plus de 300 caractères est plus facile à traiter qu'un texte de taille inférieure. Pour cette raison, nous avons essayé dans notre travail de maximiser les performances des algorithmes avec des segments de texte plutôt court jusqu'à un mot de trois caractères. La tâche peut être modélisée comme un problème de classification et est basée sur l'apprentissage machine ou bien les approches linguistique.

Pour atteindre l'objectif cité précédemment, nous avons organisé notre mémoire autour de cinq chapitres :

- **Un premier chapitre «Identification de la langue »** : dans lequel, nous présentons un aperçu sur la notion de l'Identification de la Langue (LID) ainsi que son historique, ensuite nous définissons les différents domaines dans lesquels l'identification de la langue intervient ainsi que les défis et les caractéristiques de LID.
- **Un deuxième chapitre « La classification de texte» :** dans lequel, nous allons définir la notion de classification et son intérêt dans le domaine de l'Identification de la langue. Ensuite, nous allons présenter les différentes applications de classification pour finir avec les problèmes ou les difficultés spécifiques aux textes lors de l'apprentissage automatique.
- **Un troisième chapitre « Les approches de l'identification**» : dans lequel, nous exposons les méthodes de classification (statistique et linguistique) avec les avantages et les limites de chaque méthode.
- **Un quatrième chapitre « Conception et modélisation de système » :** dans ce chapitre nous présentons la conception de nos solutions proposées.
- **Un Cinquième chapitre « Implémentation et expérimentation »** : dans lequel nous présentons les résultats obtenus suite aux tests que nous avons menés pour résoudre le problème de Détection de la Langue ainsi qu'une discussion et pour finaliser le travail nous réalisons une application web.

Chapitre 01

Identification de la Langue

CHAPITRE 1 : IDENTIFICATION DE LA LANGUE

1. Introduction

Les travaux sur L'Identification Automatique de la Langue remontent aux recherches fondamentales de Beesley (1988) [1], Cavnar et Trenkle (1994) [2]. Bien que la méthode LID puisse être extrêmement précise pour distinguer des langues qui utilisent des jeux de caractères distincts (par exemple, le chinois ou le japonais) ou qui sont très dissemblables (par exemple, persan et urdu), ses performances se dégradent lorsqu'elle est utilisée pour distinguer des langues ou des dialectes similaires.

Dans ce chapitre nous présentons d'abord l'Identification de la Langue et les travaux effectués sur l'identification de la langue, leurs domaines et les défis de LID, Nous exposons par la suite les caractéristiques de la LID et enfin nous citons ses approches.

2. Définition de l'Identification de la Langue

L'Identification de la Langue(LID) (aussi appelée reconnaissance de la langue ou La détermination automatique de la langue) est la tâche de déterminer la langue d'un texte électronique donné, qui peut être au niveau du document, du sous-document ou même de la phrase parmi un ensemble fini des langues. Bien que cette tâche soit généralement considérée comme un problème résolu pour les documents, on s'est récemment penché sur la discrimination entre les langues proches ou les variantes de langues dans diverses applications relevant du domaine Traitement Automatique de la Langue (TAL). A cet effet, la LID peut s'avérer une tâche de prétraitement importante en : Recherche d'Information, Correction Orthographique, Traduction Automatique, ou encore dans les discussions des agents conversationnels.

La LID peut être réalisée avec succès avec près de 100% de précision lorsque la longueur du texte dépasse 300 caractères (La longueur du texte est directement proportionnelle à la précision, la précision diminue lorsque la taille de texte diminue). Cette étude porte sur la LID de mots individuels, qui est toujours considérée comme une tâche complexe [3] .

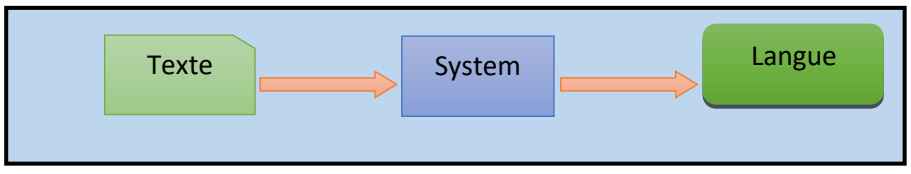


Figure 1: Processus d'Identification de la Langue

3. Travaux connexes sur l'Identification des Langues

Le premier intérêt dans ce domaine a été motivé par les besoins des traducteurs basés sur des méthodes manuelles simples. A la fin des années 60, Mustonen [4] un statisticien a compilé une liste des caractéristiques linguistiques des caractères (l'anglais, le suédois et le finnois) et en 1967 Gold [5]a examiné l'identification des langues comme une tâche dans la théorie des automates, Puis Leonard et Doddington [6] étaient capables de reconnaître cinq langues différentes en 1974.

Dans les années 80, Beesley [1]a suggéré d'utiliser des techniques crypto analytiques. Des modèles à motivation linguistique pour identification de la langue ont également été proposés, tels que le chevauchement des listes de mots vides par Johnson, 1993 [7], où un document est classé en fonction de son chevauchement avec des listes pour différentes langue.

Mots fréquents un ensemble de mots ayant la fréquence la plus élevée de tous les mots présents dans un texte Souter et al. 1994 ont pris en compte cent mots à haute fréquence par langue extraits des données d'apprentissage pour neuf langues et Grefenstette en 1995 [8]a utilisé un million de caractères de texte pour chaque langue, en les tokenisant et en extrayant tous les mots d'une longueur maximale de cinq caractères qui se sont produits au moins trois fois (les mots courts) Quant à Sibun et Reynar [9]ont introduit une méthode de détection des langues basée sur l'entropie relative, une mesure populaire également connue sous le nom de distance Kullback-Leibler. L'entropie relative est une mesure utile de la similarité entre les distributions de probabilité.

Depuis les années 1990, l'identification de la langue a été formulée comme une tâche d'apprentissage machine supervisée, et a été fortement influencé par la catégorisation des textes en général. Puis MacNamara et ses collaborateurs 1998 ont comparé un classificateur NB à un classificateur de réseau neuronal.

Linguini, un système proposé par Prager, [10], combine les modèles basés sur les mots et les n-grammes en utilisant un modèle basé sur l'espace vectoriel et examine l'efficacité du modèle combiné et des caractéristiques individuelles sur des textes de taille variable. Suzuki et al. 2002 [11] ont utilisé une méthode basée sur les n-grammes qui était capable d'identifier ensemble la langue et les schémas d'encodage des caractères pour un document

web ,Xafopoulos et ses collaborateurs 2004 ont proposé un système basé sur les modèles de Markov cachés (HMM) pour permettre la modélisation des séquences de caractères, Padro (2004) [12] a constaté que pour des langues similaires, la précision de la classification diminue .

Hammarstrom, 2007 [13]a proposé un modèle utilisant un dictionnaire de fréquence et des informations d'apposition afin d'identifier la langue de textes aussi courts qu'un mot. Ng et Selamat 2009 [14] ont procédé à l'identification de la langue sur des textes arabes en utilisant des lettres comme caractéristiques et se combinent avec les réseaux de neurones, Gottron et Lipka 2010 [15] ont comparé les différentes approches pour l'identification de la langue liée aux requêtes. Ils ont utilisé des modèles naïfs de Bayes, de la chaîne de Markov, du rang de fréquence et de l'espace vectoriel.

Winkelmolen et Mascardi en 2011 [16] ont appliqué des méthodes statistiques pour l'identification de la langue dans des documents courts tels que les textes de SMS.

Ces dernières années, les techniques de classification standard telle que les machines à vecteurs de support sont également devenues populaires et de nombreux chercheurs les ont utilisées Kruengkrai et al. 2005 [17]ou Baldwin et Lui 2010 pour identifier les langues, M. Lui et T. Baldwin 2011 [18]ont utilisé un classificateur multinomial de Bayes naïf avec une sélection de caractéristiques n-gram sur plusieurs domaines et aussi des méthodes non supervisées aussi Zhai et ses collaborateurs 2006 [19] ont montré qu'un modèle SVM à n-grammes est plus performant qu'un modèle NB à n-grammes. Amine et al. 2010 [20]ont utilisé un ensemble des algorithmes de fourmis artificielles et de k-means pour l'identification des langues.

Récemment plusieurs chercheurs ont étudié l'identification de la langue dans le contexte de langues confuses, notamment le malaisien-indonésien (Bali 2006), [21] le farsi-dari (Malmasi et Dras 2015) [22], le croate-slovène-serbe (Ljubesic et al. 2007) [23]et des variétés chinoises (Huang et Lee 2008) [24] .

4. Les domaines d'Identification de la Langue

L'identification de la langue est un élément clé dans nombreux systèmes de traitement de texte. Elle est utilisée généralement comme la première étape dans la plupart des tâches de TAL.

Notre époque est une ère de communication multilingue, qu'il s'agisse de communications

entre humains ou entre humains et machines. Ce constat implique le développement d'applications capables de gérer plusieurs langues et/ou d'identifier une langue parmi d'autres. Il est de plus en plus nécessaire aujourd'hui de traiter des documents multilingues. Si nous pouvons segmenter les documents multilingues par langue.

Il existe plusieurs applications importantes pour l'identification automatique de la langue. Nous en citons quelques domaines ci-dessous :

- Systèmes de Dialogue Multilingue : le système a besoin de savoir dans quelle langue le texte est écrit, pour pouvoir comprendre et répondre à l'utilisateur. On peut distinguer deux approches possibles pour déterminer la langue et dialoguer dans celle-ci : exécuter en parallèle autant de systèmes de reconnaissance que de langues traitées par le système de dialogue, ou bien utiliser un système dédié à l'identification de la langue, qui permet de lister rapidement les langues les plus probables qui sont ensuite départagées par les systèmes de reconnaissance de la parole adaptés aux langues présélectionnées. La dernière approche, en considérant les contraintes de temps-réel d'un tel système, permet la prise en compte d'un nombre bien plus important de langues [25].
- Les Requêtes des Moteurs de Recherche : Le problème de l'Identification de la Langue des textes écrits prend de plus en plus d'importance car plusieurs moteurs de recherche automatiques utilisent des informations linguistiques pour répondre aux exigences des requêtes. De plus, il existe de nombreux documents qu'il faut classer selon des critères donc l'identification correcte de la langue d'une requête est primordiale afin de permettre au moteur de recherche de fournir les résultats les plus pertinents [26] .Par ailleurs, la détection de la langue est de nos jours, utilisée dans les navigateurs web pour préfiltrer les résultats de recherche afin que les utilisateurs n'aient pas à parcourir des milliers de pages qu'ils ne comprennent pas. Les principaux moteurs de recherche tels que Google[1] et Bing[2] parcourent des milliards de pages web dans plus de 50 langues.
- Pour la Recherche d'Informations : L'Identification de la Langue est très utile car les utilisateurs peuvent être aidés à filtrer les documents en fonction de la langue choisie. Tels que chaque document est d'abord classé dans la langue écrite qui lui appartient et est ensuite classé en fonction de la langue choisie.

[1] http://www.google.com
[2] http://www.bing.com

Nous constatons également que Détecter automatiquement la langue disponible dans plusieurs outils comme Word et Outlook sous Windows.

- L'analyse textuelle des médias sociaux : est rapidement devenue l'un des domaines "frontières" du Traitement Automatique de la Langue(TAL). les grandes conférences lui ayant ouvert des voies entières ces dernières années. Les défis que pose le TAL aux médias sociaux sont nombreux, et découlent principalement de la nature "bruyante" du contenu [27] et donc de la langue.
- Traitement Automatique de la Langue (TAL) : il existe des applications de Traitement Automatique de la Langue (TAL) qui ont besoin l'information sur la langue d'un document qui est une condition préalable nécessaire à la poursuite des étapes de traitement, la catégorisation des documents, correcteurs orthographiques, le résumé et la traduction automatique sont tous dépend de la connaissance de la langue du document.

5. Les défis de l'Identification de la Langue

Identification de la langue est un domaine très important et nécessaire au fil de temps l'identification a reconnu plusieurs travaux mais nous avons observé que les chercheurs ont confrontés à des difficultés courantes, dans la section suivante nous allons résumer quelques défis de l'identification de la langue :

5.1. Les langues proches

En 1995 Grefenstette [8] a constaté qu'il y avait une confusion de classification entre les langues de la famille germanique (Néerlandais et Allemand) et les langues de la famille latine (Espagnol et Portugais).

Nous constatons qu'il est beaucoup plus difficile de discriminer les langues au sein des familles phylogénétiques [3] que les langues des différentes familles Tchèques et Slovaque, Anglais Américain et Anglais Britannique alors que les langues Persanes et Urdu sont toujours ouvertes pour la recherche.

5.2. La différence entre une langue et un dialecte

Une langue est souvent associée à une nation ou à une réalité géopolitique. Un dialecte, quant à lui, il peut être considéré comme une langue mais qui est parlée

[3] Phylogénétiques : des langues de même origine de langues

par un groupe restreint de personnes donc un dialecte est un sous-ensemble d'une langue. Par conséquent, la machine peut trouver des difficultés à les différencier.

5.3. La taille du fragment de texte à identifier

Dans de nombreuses applications, il est souhaitable d'identifier la langue à partir d'une quantité limitée de texte comme les messages courts (SMS) de moins de dix mots.

5.4. La quantité et la variété des données disponibles sur l'apprentissage

Dans de nombreuses applications de TAL, la taille des données de l'apprentissage disponibles influence les performances globales du système, comme l'a montré G. Selon Hidayet Takçı, [28] dans les études d'identification des langues les corpus disponible prés de million de mots.

5.5. L'algorithme de classification utilisé

Plusieurs algorithmes sont potentiellement applicables à la classification mais dans l'identification de la langue les différences de langues peuvent se réduire, de sorte que le classificateur sera forcé d'apprendre des différences mineures et perdra sa capacité à généralisation [29].

6. Caractéristiques de l'Identification de Langue

Pour déterminer la langue dans laquelle un document d'entrée est rédigé, la décision est prise en fonction des caractéristiques du document, généralement au « niveau du mot » ou du « caractère ». Selon Tommi Jauhiainen et al. [30], la partie la plus importante d'un travail d'Identification de la Langue consiste à trouver des fonctionnalités qui peuvent être utilisées pour aider à la tâche. Certaines de ses caractéristiques sont résumées dans ce qui suit [30]:

6.1. Octets et codages

Les documents sont numérisés en utilisant un codage particulier, qui transforme les caractères d'alphabet en séquence d'octet qui peut être stockée par la suite dans l'ordinateur, Certains codages sont spécifiques à une langue donnée (par exemple GuoBiao 18030 ou Big5 pour chinois et Shift-JIS pour le japonais), et d'autres comme la famille de codages Unicode sont spécifiquement conçus pour représenter le plus grand nombre de langues possible.

En 1996 Kikui [31] a introduit une étape de détection de l'encodage dans le traitement de LID mais vu qu'il était coûteux en termes de calcul, plusieurs chercheurs comme Mandl, Shramko,Tartakovski, & Womser-Hacker en 2006 [32]ont conclus à la possibilité d'ignorer l'encodage et partent du fait que tous les documents utilisent le même encodage par exemple toutes les données de Twitter et de Wikipédia sont codées en UTF-8.

6.2. Caractères

- **Caractères non alphabétiques ou non idéographiques**

Simaki, Simakis, Paradis et Kerren en 2017 ont utilisé les fréquences de ponctuation et certains symboles spéciaux comme caractéristiques pour distinguer les variantes anglaises [33].

- **Alphabets**

En 1989 Henrich [34] a utilisé la connaissance des alphabets pour exclure les langues dans lesquelles un caractère propre à une langue n'apparaît pas dans un document de test.

- **Capitalisation**

Parfois la capitalisation peut aider surtout le calcul des fréquences des caractères n-grammes mais dans le machine Learning quand on utilise l'orthographe d'un document on préfère la minuscule car il peut se produire une ambiguïté) par exemple : machine et MACHINE ne sont pas équivalents.

- **Le nombre de caractères dans les mots**

Langer en 2001 [35] a été le premier qui a utilisé la longueur de mot pour l'identification de la langue, la méthode basée sur les mots est utilisée pour identifier les langues qui marquent les limites des mots comme par exemple le nombre moyen de lettres par mot en français varie entre 5 et 6 lettres par contre en allemand La taille du mot peut être égale 36.

- **La fréquence de chaque caractère**

Kerwin (2006) a utilisé les fréquences des caractères comme vecteurs caractéristiques. Par exemple la phrase : « L'identification de la langue d'un texte est utile dans un large éventail d'applications » contient 73 lettres dont 7 sont des "L", ce qui fait que la fréquence relative à cette lettre est de 0,10, soit 10 %. On se base ensuite sur ces fréquences pour détecter la langue. La figure 2 représente la fréquence relative de lettre « o » dans diverses langues [36].on peut aussi utiliser la probabilité des caractères.

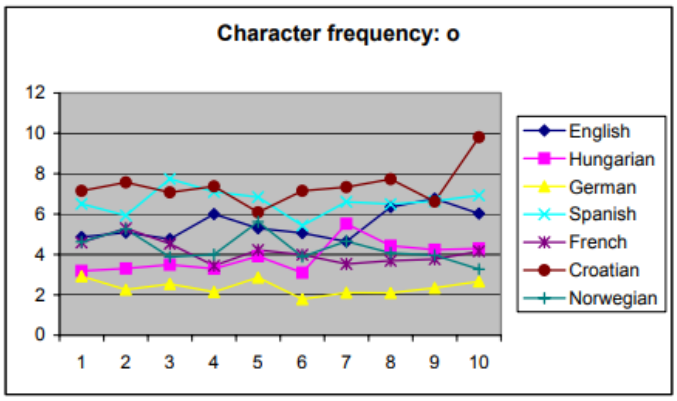


Figure 2 : la fréquence relative de lettre « o »

6.3. Combinaisons de caractères

- **Relation voyelle-consonne**

Rau (1974) [37] a utilisé les fréquences relatives des voyelles qui suivent les voyelles, des consonnes qui suivent les voyelles, des voyelles qui suivent les consonnes et des consonnes qui suivent les consonnes.

- **Répétition des caractères**

Banerjee et al. [38] Ont utilisé la présence de caractères se répétant plus de deux fois en 2014.

- **Des n-grammes de caractères de tailles différentes**

Väyrynen, et Virpioja [39] en 2010 ont créé un identificateur de langue basé sur des n-grammes de caractères de tailles différentes sur 281 langues, et ont obtenu une précision d'identification de 62,8 % pour des échantillons extrêmement courts (5-9 caractères).

- **Séquences de consonnes ou de voyelles**

Nombre de fois deux ou trois consonnes se trouvent entre deux voyelles dans un mot.

Les consonnes simples et doubles séparent les langues très bien. Le fait d'avoir une paire de consonnes est moins fréquent, il varie de 10 à 20% selon la langue mais les valeurs sont quasi constante pour une langue donnée, cette fonction est donc utile [36].

Sterneberg [40] a utilisé des séquences de consonnes générées à partir de mots comme l'une des caractéristiques d'un classificateur SVM en 2012.

6.4. Morphèmes, syllabes et morceaux

- **Morphèmes**

En linguistique un morphème est définit comme le plus petit élément significatif (préfix, suffixe, affixe). Par exemple, le mot chanteurs est composé de trois morphèmes : chant- « chant », -eur- « celui qui fait » et -s (marque du pluriel à l'écrit seulement).

En 2005 Marcadet, Fischer, et Waast-Richard [41] ont utilisé des suffixes et des préfixes dans l'Identification de la Langue basée sur des règles.

- **Syllabes et n-grammes de syllabes**

Une syllabe est une unité de prononciation comportant une seule voyelle, avec ou sans consonnes environnantes, formant la totalité ou une partie d'un mot. Chen, You, Chu, Zhao et Wang [42] ont utilisé des trigrammes composés de syllabes pour la détection de la langue en 2006.

- **Morceaux, n-grammes de morceaux**

Les morceaux représentent certaines combinaisons de caractères qui sont considérées comme ayant une signification particulière dans une langue donnée. You et al. [43] Ont utilisé les trigrammes de morceaux non syllabes 2008.

6.5. Mots

- **Dictionnaire de base**

Les dictionnaires des bases sont des listes nettoyées et non ordonnées de mots et ne conservent qu'une seule occurrence pour chaque mot. Ils permettent de supprimer toutes les ambiguïtés mots : mots qui apparaissent dans plus d'une langue. Le rôle des dictionnaires est considéré comme des réserves d'exceptions et a été utilisé pour la LID par Adouane et Dobnik (2017) [44].

- **Mots discriminants**

En utilisant des mots uniques ou presque uniques comme les mots vides, Rehůřek et Kolkus [45] en 2009 ont utilisé les mots spécifiques les plus pertinents pour chaque langue, Cette méthode est utilisée en l'Identification des Langues sur le Web.

- **Mots courts**

Appelée méthode des petits mots, généralement moins de quatre lettres sont extraites et qui peuvent représenter des adjectifs, prépositions qui apparaissent dans chaque document d'une langue donnée utilisée par Grefenstette en 1995 [8].

6.6. Combinaisons de mots

- **Longueur de la phrase**

Les longueurs de phrases sont considérées comme « Meta Feature »[4] qui ne sont pas directement liées à la dialectique des mots dans la phrase donnée mais plutôt pour estimer le caractère informel de la phrase et d'inclure comme Elfardy et Diab [46]ont utilisé le nombre de mots dans une phrase avec Naïve Bayésienne(NB) en 2013.

- **Mot n-grammes**

Singh [47] 2006 a utilisé des trigrammes de mots simultanément avec le caractère 4-grammes. Il a conclu que les modèles basés sur les mots peuvent être utilisés pour augmenter les résultats des n-grammes de caractères lorsqu'ils ne fournissent pas de résultats d'identification fiables.

- **Syntaxe et part-of-speech (“POS”)-tags**

Martinc et al. (2017) ont utilisé des trigrammes de POS-tag (Un tagger de partie de discours (PoS) qui étiquettent les mots comme l'une des différentes catégories pour identifier la fonction du mot dans une langue donnée en tant que noms , verbes , adjectifs , adverbes , etc.) avec pondération TF-IDF [48].

7. Les approches de l’Identification de la Langue

On trouve en principe deux approches différentes pour l'identification automatique de la langue d'un document électronique :

7.1.Identification linguistique

Approche basée sur les ressources linguistiques comme les mots, caractères ….En général les identificateurs linguistiques fonctionnent de manière très fiable pour les documents

[4] Meta Feature : Le préfixe «méta» est d'origine grecque et signifie «parmi», «après», «à côté de» ou «avec». En tant que préfixe, il est souvent utilisé pour identifier quelque chose qui fournit des informations sur autre chose.

longs et de taille moyenne. Pour les documents de plus de 50 octets de texte, l'identifiant présenté fonctionne avec un rappel d'environ 96 % et une précision proche de 100 %.

Elle ne peut pas être utilisée pour les langues qui ne marquent pas systématiquement les limites des mots par des blancs ou des signes de ponctuation, comme le japonais, le chinois et le coréen. Pour marquer un morceau de texte dans ces langues, la langue doit être connue au préalable.

- Il s'agit de modéliser une certaine partie de la connaissance linguistique afin de la rendre exploitable par la machine.
- Dépendantes de la langue.
- Plus difficiles à mettre en place : nécessitent une conceptualisation des phénomènes linguistiques.
- Nécessitent plus de temps et plus de travail (par des linguistes).
- Les résultats du système, notamment les erreurs, peuvent être expliqués et corrigés facilement.

7.2.Identifications Statistiques

- elles s'appuient sur un formalisme mathématique.
- Applicables à des corpus de très grande taille.
- Indépendantes de la langue.
- Ne nécessitent pas de connaissances linguistiques.
- Ne permettent pas de comprendre les phénomènes linguistiques.
- Méthodes : n-grammes, apprentissage automatique.
- Les résultats du système, notamment les erreurs, sont difficiles à expliquer et corriger.

8. Conclusion

L'étude des langues et des mécanismes nécessaires à la mise en œuvre à son traitement automatique par des machines est un domaine d'études foisonnant, parmi ce domaine l'identification de la langue.

A travers ce premier chapitre, nous avons mis le point sur les concepts de base de l'identification de la langue, nous avons ainsi exploré les principales générations des produits et systèmes TAL.

Dans ce qui suit, nous allons mettre le point sur la classification de textes comme technique pour l'Identification de la Langue.

Chapitre 02

La classification de texte

CHAPITRE 2 : LA CLASSIFICATION DE TEXTE

1. Introduction

L'identification de la langue parmi les différentes applications de la catégorisation de texte qui est une tâche générique est consisté à assigner une ou plusieurs catégories parmi une liste prédéfinie a connu un intérêt croissant au cours des dix dernières années, en raison de la disponibilité accrue des documents sous forme numérique et de la nécessité qui en découle de les organiser [49].

Les documents textuels sont classés par la méthode d'identification de la langue basée sur les catégories de langue, Elle peut être considérée comme un exemple spécifique du problème plus général de la classification, l'identification des langues est considéré comme une forme de classification des textes depuis les années 1990 [50].

La Catégorisation de textes (CT) est aujourd'hui un domaine de recherche bien établi et très actif. Les travaux portent sur les systèmes avec apprentissage des catégories à partir de corpus pré-étiquetés. Dans ce chapitre, nous présentons d'abord une définition de la catégorisation des textes, l'objectif de classification et l'application de CT ainsi que le processus de CT qui est constitué de plusieurs étapes : la collection de documents, prétraitement des textes, la pondération des termes et le choix de classificateur, évaluation du processus de catégorisation (rappel, précision,..). Nous exposons par la suite les applications de la CT et enfin nous citons les principales difficultés liées à la CT.

2. Définition de Classification

La classification automatique de texte (aussi appelée catégorisation de texte ou balisage de texte) est un problème connu en informatique, C'est l'une des tâches fondamentales du traitement automatique de la langue (TAL), il s'agit d'assigner un document à une ou plusieurs catégories ou classes. Le problème est différent selon la nature des documents en question, en effet la classification de textes diffère de la classification de documents images, vidéo ou encore son. On peut aussi imaginer des classifications selon des paramètres associés aux documents tels que par exemple l'auteur, la langue et etc. [51] .

La catégorisions des textes (étiquettes, classes) consiste à chercher une liaison fonctionnelle entre un ensemble de textes et un ensemble de catégories cette liaison fonctionnelle que l'on appelle également modèle de prédiction, est estimée par un

apprentissage automatique (Machine Learning en Anglais). Pour se faire, il est nécessaire de disposer d'un ensemble de textes préalablement étiquetés, dit ensemble d'apprentissage.

Formellement, un processus de catégorisation se définit comme une fonction Φ telle que :

$$\Phi : D \times C \{V, F\}$$

Telle que la valeur V (vrai) est associer au couple (D_I, C_J) si le texte du document D_i appartient à la catégorie C_j, et F (faux) est associé au couple dans le cas contraire.

3. Objectifs de la classification

L'objectif de la classification de textes est de rassembler les textes similaires selon un certain critère, au sein d'une même classe. Le but étant d'organiser, structurer et catégoriser à peu près n'importe quoi. Par exemple, les nouveaux articles peuvent être organisés par thèmes, les tickets de support peuvent être organisés par urgence, les documents peuvent êtres organiser selon la langue, les mentions de marque peuvent être organisées par sentiment, etc….Le principal avantage des classificateurs est leur haute performance à la fois au stade de l'apprentissage et au stade de la classification.

4. Applications de la classification de textes

Dans la première histoire de la Machine Learning (ML) et de l'Intelligence Artificielle (IA), les techniques de classification de texte ont été principalement utilisées pour systèmes de recherche d'informations. Cependant, comme les progrès technologiques ont émergé au fil du temps, la classification de texte et la catégorisation des documents ont été utilisées à l'échelle mondiale dans de nombreux domaines tels que identification de la langue, la médecine, les sciences sociales, la santé, la psychologie, le droit, l'ingénierie, etc... Dans cette section, nous mettons en évidence plusieurs domaines utilisant des techniques de classification de texte [52] .Nous avons mentionné et expliqué certaines applications :

Tableau 1: Exemples des applications de classification

Les applications	Description	Classification populaire
Identification de la langue	On peut considère LID comme un problème de classification par langue	SVM
La recherche d'informations (RI)	On s'intéressera en particulier à la lente émergence de la problématique textuelle qui accompagne l'expansion du Web et pour facilite la tâche de RI nous avons besoin de la classification	Naïve Bayes, SVM, décision arbre, KNN
Filtrage d'information	Le filtrage de l'information consiste à sélectionner les informations pertinentes ou à rejeter les informations non pertinentes des informations provenant d'un flux de données entrant.	Naïve bayésien, SVM
Analyse des sentiments	Les méthodes de classification des sentiments classent un document associé à une opinion positive ou négative.	Bayésien naïf et SVM
Résumé des documents	Un résumé est également nécessaire en raison de l'augmentation rapide des informations en ligne et pour extraire des caractéristiques importantes d'un document.	Naïve Bayes C4.5

5. Processus de classification

Le processus de catégorisation est un système (figure 3) qui reçoit en entrée un texte et en sortie associe une ou plusieurs catégories. Ceci est effectué en respectant un ensemble d'étapes. Ces étapes concernent le choix de collection des documents et le prétraitement, la représentation des textes, le choix de l'algorithme d'apprentissage, et l'évaluation des résultats en vue de prévoir le degré de généralisation du classifier.

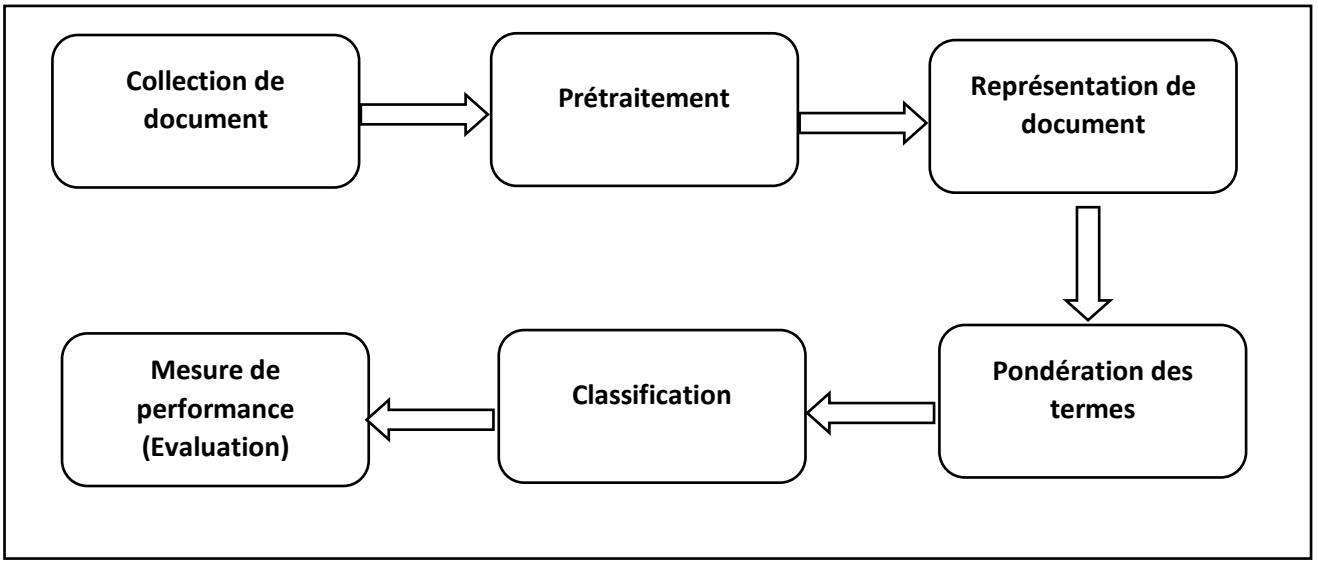


Figure 3 : Processus de classification

Dans ce qui suit, nous allons décrire ces différentes étapes :

5.1.Collection de document

Le corpus (en anglais, « data set ») est la première étape pour le lancement du processus de classification. C'est un élément essentiel à la construction d'un système de classification automatique. Il existe des corpus payant et des corpus open source qui bien structurés dans plusieurs sites web, on peut utiliser de format d'origine ou modifier le format selon les besoins (CSV, TXT, XLSX...).

5.2.Prétraitement (Preprocessing)

La plupart des ensembles de données de texte et de document contiennent de nombreux mots inutiles tels que des mots vides, faute d'orthographe, etc. Dans de nombreux algorithmes, en particulier les algorithmes d'apprentissage statistiques et probabilistes, le bruit et les fonctionnalités inutiles peuvent avoir des effets néfastes sur les performances du système. Dans cette section, nous expliquons brièvement quelques techniques et méthodes de nettoyage de texte et de prétraitement des ensembles de données textuelles.

5.2.1. Ponctuation

La ponctuation a pour but l'organisation de l'écrit grâce à un ensemble de signes graphiques comme (".","";",,:","!","?"), un seul signe de ponctuation peut modifier la nature d'une phrase ,la ponctuation à faciliter la compréhension du texte, elle est un

élément essentiel de la communication écrite, mais a cote de traitement automatique de la langue aucun algorithme comprendre ces signe donc convenu à enlever les caractères spéciaux et les signes de ponctuations et des chiffres et etc.…

5.2.2. Tokenization

Un Token est une unité définie comme une séquence de caractères, La Tokenization est une méthode de prétraitement qui divise le texte, se produit à différents niveaux : un texte peut être divisé en paragraphes, phrases, mots, symboles ou phonèmes.

Exemple :

Après avoir dormi pendant quatre heures, il a décidé de dormir encore quatre heures/

Dans ce cas, le résultat de Tokenization par mot sera :

{“Après”, “avoir “, “dormir ”, “ pendant “,”quatre ”, “ heures ”, ”il ”, ”a ”, “décidé ”, “ de ” , “dormir ” ,“ encore ”, “ quatre” ,“ heures ”}.

5.2.3. Elimination des mots vides

Une fois les documents textes découpés en Token, nous apercevons que certains de ces Token sont présents dans tous les textes du corpus, c'est ce que nous appelons les mots vides (stops Word en anglais) être : les prépositions, les mots de liaisons, les déterminants, les adverbes, les adjectifs indéfinis, les conjonctions, les pronoms et les verbes auxiliaires etc… comme "la ,le , dans, car, les " dans la langue française, et "the, and, to,bay,after,of dans la langue anglaise . (,نام, اول, تواند, هايي, گرفته, روي, بسيار) dans la langue persan , (خبًب اچھے, ضلطلہ, دیر, درخبت, چلے,) dans la langue ourdou . Qui représente une grande part des mots d’un texte, mais malheureusement sont faiblement informatifs, sur le sens d’un texte puisqu’ils sont présents sur l’ensemble des textes. C’est mot ne contiennent pas aucune information sémantique, qui ne modifie pas le sens des mots.

La classification des textes et des documents comprennent bon nombre de ces mots qui ne contiennent pas signification à utiliser dans les algorithmes de classification. La technique la plus courante pour traiter ces mots est de les supprimer des textes et documents [53].leur suppression réduit la taille du lexique du document, le temps de traitement et le temps d'apprentissage seront réduit considérablement.

5.3.Représentation de texte

À chaque fois qu'il est question de définir un problème de façon à assurer un traitement automatique, il est impossible de passer outre l'étape où il faut choisir la façon dont on va représenter le problème. Dans le cas de la classification automatique de textes Les algorithmes d'apprentissage ne sont pas capables de traiter directement les textes, on doit opter pour une façon efficace de représenter les instances à traiter, soit les textes.

Cette étape consiste généralement en la représentation de chaque document par un vecteur, dont les composantes sont par exemple les mots contenus dans le texte, afin de le rendre exploitable par les algorithmes d'apprentissage.

5.3.1. Représentation en Sac de mot (bag of Word)

Le modèle du sac de mots est une représentation réduite et simplifiée d'un texte, La technique de sac de mots est utilisée dans plusieurs domaines tels que la vision par ordinateur, TAL, le spam bayésien ainsi que la classification des documents et la recherche d'informations par apprentissage automatique. Cette représentation des textes exclut toute analyse grammaticale et toute notion de distance entre les mots.

Un grand nombre d'auteurs comme Lewis en 1992 [54] utilisent les mots comme composantes du vecteur pour représenter les textes .Cette représentation il s'agit de transformés simplement les textes en vecteurs dont chaque composante représente un mot.

Exemple :

Document : Soyer optimistes et tout ira pour le mieux.

Sac de mots :

{"Soyer ", "optimistes ", "et ", " tout ","ira ", " pour ", "le ", "mieux"}.

5.3.2. Représentation en vecteur binaire

Certains algorithmes de machine Learning Naïve Bayes nécessitent l'utilisation de valeurs binaires. Il sera dès lors nécessaire de définir une certaine valeur «seuil» d'occurrence, une cellule w_{ij} sera considérée comme vraie (1) ou fausse (0). Ce qui correspond simplement à la présence ou l'absence d'un terme dans un document [55].

Exemple :

D1 : le chien bleu a mangé un biscuit bleu

D2 : le chien bleu a mangé un biscuit rouge

Tableau 2: Représentation binaire

	Chien	Bleu	mangé	Biscuit	Bleu	rouge
D1	1	1	1	1	1	0
D2	1	1	1	1	0	1

5.3.3. Représentions Par des phrases

Un Certains nombres des chercheurs comme S. Marvin and S. Scott [56] proposent la phrase comme unité de représentation car parfois les phrases peuvent être plus informatives que les mots. Par exemple «traitement automatique de la langue» et « recherche informatique » car les phrases ont l'avantage de conserver l'information relative à la position du mot dans la phrase et possède un degré d'ambiguïté plus petit.

5.3.4. Méthode de n-gramme

Dans la littérature, ce terme désigne quelque fois des séquences qui ne sont ni ordonnées ni consécutives ; par exemple un bigramme peut être composé de la première lettre et de la troisième lettre d'un mot Cavnar and Trenkle 1994 [2], de nombreux travaux ont montré l'efficacité des n-grammes (n variant de 1 à 5) comme méthode de représentation des textes pour leur catégorisation .L'avantage est qu'il n'est pas nécessaire de rassembler des connaissances linguistiques pour construire un classificateur. Les n-grammes sont également extrêmement simples à calculer pour un texte donné.

Les n-grammes de longueur 1, 2 et 3 sont généralement appelés respectivement unigrammes, bigrammes et trigrammes ; pour les longueurs $n \geq 4$, on utilise le terme "N-gram".

Ils peuvent être consécutifs ou se chevaucher. Les bigrammes de caractères consécutifs créés à partir de «corpus »*la* séquence de six caractères sont co et rp tandis que les bigrammes qui se chevauchent sont co, or, et pr... Les n-grammes qui se chevauchent sont le plus souvent utilisés dans la littérature.

Par exemple, la décomposition du mot "corpus" donne le résultat suivant :

Tableau 3 : Représentation classique en n-gram du mot "corpus".

n = 1	n = 2	n = 3	n = 4	n = 5
_	_c	_co	_cor	_corp
c	co	Cor	corp	corpu
o	or	Orp	orpu	orpus
r	rp	rpu	rpus	rpus_
p	pu	pus	pus_	pus__
u	us	us_	us__	us___
s	s_	s__	s___	s____

5.3.5. Représentation par Racinnisation

En anglais « stemmer » est étape principale utilisée pour traiter les données textuelles dans le traitement automatique de la langue (TAL) c'est-à-dire vise à garder la racine du mot, tronquer toute déclinaison accord (flexion) et dérivations. C'est une tâche principale de la normalisation des mots qui consiste à réduire chaque mot à sa racine lexicale [57].

Cette méthode consiste à remplacer les mots du document par leurs racines lexicales, et à regrouper les mots de la même racine dans une seule composante. Ainsi, plusieurs mots du document seront remplacés par la même racine, cette méthode peut être réalisée en utilisant un des algorithmes les plus connus pour la langue anglaise qui est l'algorithme de Porter[5] La normalisation de mots sert à supprimer les affixes de ces derniers pour obtenir une forme canonique. Néanmoins la transformation automatique d'un mot à sa racine lexicale

[5] M. F. Porter, «An algorithm for suffix stripping », Program, pp 130–137, Morgan Kaufmann Publishers Inc., 1980.

peut engendrer certaine anomalies. En effet, une racine peut être commune pour des mots qui portent des sens différents tel que les mots jour, journalier, journée ont la même racine « jour » mais se rendent à trois notions différentes, cette représentation dépond aussi de la langue utilisée.

5.4.Pondération de terme

L'étape de pondération ça aide à mesurer l'importance d'un terme dans un document, on peut calculer l'importance de terme à partir de considérations et interprétations statistiques. Pour calculer la pondération on distingue les méthodes suivantes dans le but de trouver les termes qui représentent mieux le contenu de document :

5.4.1. Fréquence des termes TF (Term frequency)

On désigne par TF la fréquence d'un mot (descripteur) dans un texte donné. C'est un calcul de fréquence très simple, mais qui s'avère efficace et pratique.TF est fréquence de terme dans un document qui prend en compte le nombre d'occurrences du terme dans le document.

$$TF_{t,d} = \frac{n_{t,d}}{N_d} \qquad (1)$$

Où $n_{t,d}$ est la fréquence d'apparition du terme t dans le document d etN_d est le nombre total des termes dans d.

Le principal inconvénient de la fréquence des termes est le fait qu'il est possible, et d'ailleurs c'est un cas très probable en pratique, qu'un terme apparait avec une fréquence assez grande dans tous les documents d'un corpus. Dans ce cas, le terme en question perd toute sa notion de discrimination relative au degré de présence. Une autre notion vient rectifier ce cas exceptionnel, nommée IDF (Inverse documents fréquences).

5.4.2. Fréquence documents inverses IDF (inverse document frequency)

Est une mesure de l'importance du terme dans l'ensemble du corpus, le nombre de documents dans lequel le terme apparaît qui prend en compte le nombre d'occurrence du terme dans le corpus, elle est calculable par :

$$IDF_t = log\left(\frac{|D|}{|\{d_j, t_j \in d_j\}|}\right) \qquad (2)$$

Ou :$|D|$ le nombre total de document dans corpus

$|\{d_j : t_j \in d_j\}|$: Le nombre de documents où le terme apparait.

Si le terme est très présent dans tout le corpus alors le rapport sera égal à 1 et IDF = 0 donc le terme est neutralisé.

5.4.3. TF-IDF «term frequency inverse document frequency»

Nous avons vu que la fréquence d'un terme dans un document joue un rôle important dans le calcul de son degré de discrimination. En revanche, la représentation d'un texte, dans le but de le classifier, ne dépend pas seulement de son contenu, mais elle est liée étroitement au corpus auquel le texte appartient, la rareté de ce terme au sein des autres documents du corpus s'avère aussi importante que sa fréquence (abondance) dans le document en question. Cette combinaison judicieuse de ces deux principes (abondance particulière et rareté générale) a engendré la pondération dite TFIDF.

Une technique d'optimisation couramment utilisée pour la catégorisation de texte et avoir des représentations plus riches en informations pour calculer le poids de w_{ij}

$$TF - IDF = TF_{t,d} \times IDF_t \qquad (3)$$

Afin d'éviter les problèmes posés par les différentes longueurs de texte, on doit avoir recours à une représentation TF-IDF normalisée nomme le codage TFC et comme par exemple le codage LTC Buckley et al .en 1994 [58]qui tente de réduire les effets des différences de fréquences.

5.5.Etape choix de classifier

L'apprentissage automatique est un sous-domaine de l'intelligence artificielle. Il vise à résoudre de façon automatique, des problèmes difficilement solubles au moyen d'algorithmes classiques C'est une méthode (mathématique, statistique) qui a été développée initialement pour la reconnaissance des images.

Le but essentiel de l'apprentissage machine est de déterminer la relation entre les objets et leurs catégories pour la prédiction et la découverte des connaissances Dans cette partie,

nous essayons de donner une vue générale sur l'apprentissage machine et ses deux déclinaisons : l'apprentissage supervisé et l'apprentissage non supervisé.

5.5.1. Apprentissage supervise

C'est approche est de classer de façon automatique les documents dans des catégories qui ont été définies soit préalablement par un expert L'idée est d'apprendre une règle de classement à partir d'un ensemble de données dont le classement est déjà connu.

L'apprentissage supervisé a pour but d'établir des règles de comportement à partir d'une base de données contenant des exemples de cas déjà étiquetés. La base de données est en principe un ensemble de couples entrées / sorties {(X, Y)}. Le but est d'apprendre à prédire pour toute nouvelle entrée X, la sortie Y. [59] L'Identification de la Langue correspond à un problème de classification supervisé, Dans la classification supervisée les langues précédemment classées forment un ensemble d'entraînement pour classer la langue. Il existe plusieurs algorithmes techniques utilisés pour la classification supervisée que nous détaillerons dans le chapitre 3.

5.5.2. Apprentissage non supervisée

L'apprentissage non supervisé consiste à apprendre à un algorithme d'intelligence artificielle(IA), les informations qui ne sont ni classées, ni étiquetées, et à permettre à cet algorithme de réagir à ces informations supervision[6]. Nous ne disposons pas non plus de données en entrée qui sont déjà classées, c'est aussi à l'algorithme de découvrir par lui-même. L'apprentissage non supervisé est utilisé dans plusieurs domaines tels que :

-Le traitement de la parole : construction de système de reconnaissance de la voie humaine.

- Traitement d'images.

- Classification de documents.

5.5.3. Les données de l'apprentissage automatique

L'ensemble des documents pré-étiquetés est séparé en deux sous-ensembles :

- L'ensemble d'apprentissage : Training Data(TD)

[6] https://www.lemagit.fr/definition/Apprentissage-non-supervise

Où on fournit à la machine un corpus d'apprentissage : un grand corpus avec des textes et des analyses correctes. La figure 4 ci-dessous illustre cette phase.

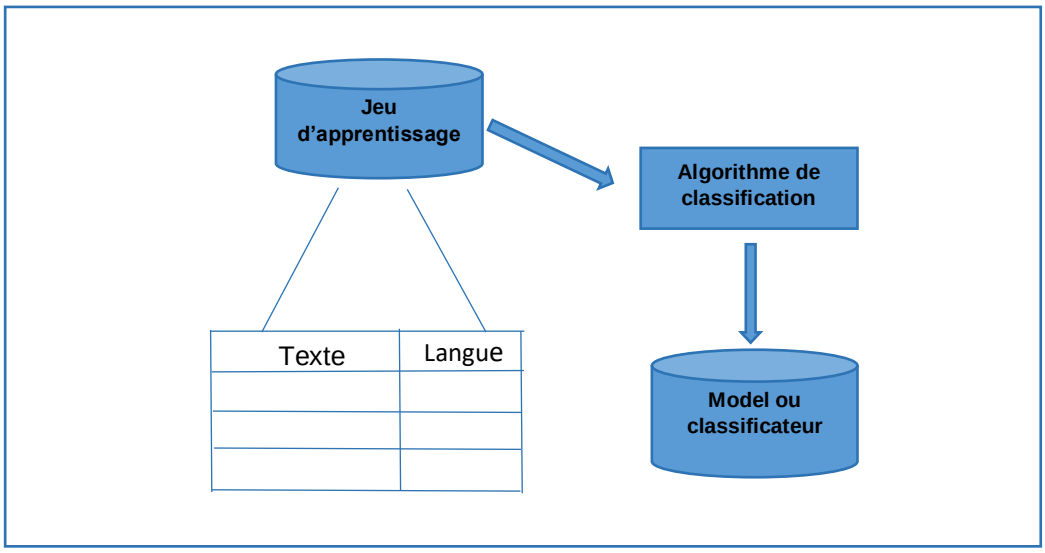


Figure 4 : Processus d'apprentissage

- L'ensemble de Test : Pendant la phase de test On teste le système sur un petit corpus de test, on soumet chaque document à l'algorithme de classification et on regarde simplement si la machine trouve la bonne classe.

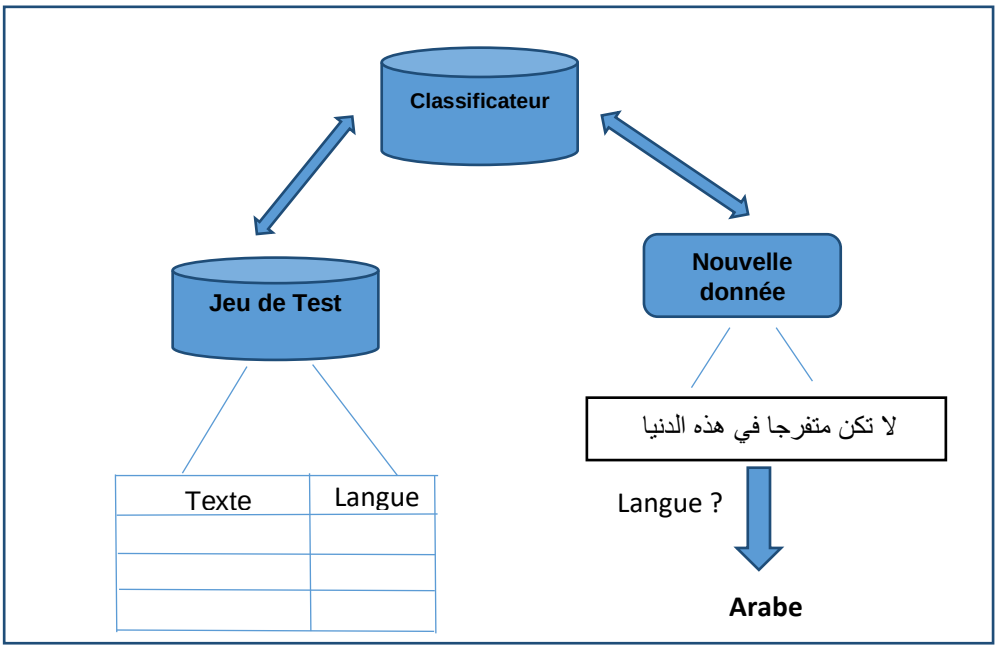


Figure 5 : Processus de test

5.5.4. Les classificateurs de l'apprentissage

Il existe de nombreuses méthodes qui sont prises en considération pour l'apprentissage des classificateurs. Certains algorithmes de classification ont été conçus en particulier pour le processus de catégorisation. Nous avons cité la topologie de quelques algorithmes :

Par analogie :

- Plus Proches Voisin (PPV).

Par combinaison de tests élémentaires :

• Arborescence :

- Arbre de Décision (AD)
- Les forêts aléatoires (FA)

• Vote pondéré : Boosting (dopage)

- Gradient Boosting machines (GBM)

Par approche probabiliste (méthodes bayésiennes)

- Bernoulli naïve Bayes (BNB)
- Multinomial naïve Bayes (MNB)

Par minimisation de l'erreur : Réseaux de neurones (MLP), etc…

Par maximisation de la « marge » :

- Algorithme Machine à vecteurs de support (SVM)
- Régression logistique (RL)

5.6.Evaluation du processus de catégorisation

Certains principes d'évaluation sont utilisés de manière courante dans le domaine de catégorisation de textes. Les performances en termes de classification sont généralement mesurées à partir de des indicateurs traditionnellement utilisés à savoir les mesures de rappel et précision. Initialement elles ont été conçues pour les systèmes de Recherche d'Information, mais par la suite la communauté de classification de textes les ont adoptées.

- **Rappel** : le pourcentage d'exemples que le classificateur a prédit pour une étiquette donnée par rapport au nombre total d'exemples qu'il aurait dû prédire pour cette balise donnée(C).

$$Rappel\ (C)\ =\ \frac{VP}{VP+FN}$$

Tel que :

VP : (True positive) c'est le nombre de documents correctement attribué a la catégorie

FN : (False négative) c'est le nombre de qui auraient dû lui être attribués mais que non l'ont pas été.

Le rappel n'est pas suffisant pour évaluer les performances d'un système. Pour cela, on définit la précision.

- **Précision :** le pourcentage d'exemples que le classificateur a obtenu (classer dans une classe C) par rapport au nombre total d'exemples qu'il a prédit pour une étiquette (classe C) donnée.

Précision (C) = $\frac{\mathrm{VP}}{\mathrm{VP+FP}}$

Tel que :

FP : (False positive) c'est le nombre de incorrectement attribués à la catégorie.

- **F_score :**c'est la moyenne harmonique de la précision et du rappel. C'est une fonction qui est maximisée quand la précision et le rappel sont proches.
- **F_messure :** est un indicateur qui combine le rappel et la précision elle est donnée par la formule suivante :

F_messure (C) = $2 * \frac{(\mathrm{rappel} * \mathrm{précision})}{\mathrm{présicion+rappel}}$

Ces mesures précédentes vont servir à évaluer le système par rapport à une seule classe. Pour une évaluation globale du classifier par rapport à toutes les classes du corpus, nous avons choisi d'utiliser les mesures micro moyenne et Macro.

- **La micro mesure** : qui correspondent à une moyenne qui pondère les classes par le nombre de documents qu'elles contiennent c'est-à-dire que tous les documents ont la même importance. La micro-moyenne accorde donc des poids importants aux catégories ayant beaucoup d'exemples.
- **Macro mesure** : c'est la moyenne des performances des classifiers pour chacune des catégories, la macro mesure privilégie la catégorie, c'est-à-dire que toutes les catégories sont d'une même importance. La macro moyenne donne un poids égal à chaque classe, tandis que la micro moyenne donne un poids égal à chaque décision de classement par document.
- **score :** le pourcentage de textes qui ont été prédits avec la bonne étiquette.

6. Les problèmes de la classification

Comme il y a beaucoup difficultés peuvent s'opposer au processus de catégorisation de textes nous avons cite ici quelques difficultés en général puis en particulier pour chaque langue.

6.1.Difficultés particulières de la classification de textes

- **Complexité de l'algorithme :** En effet, la plupart des algorithmes d'apprentissage sophistiqués, comme les arbres de décision, le k-PPV, C'est pourquoi une méthode de réduction de dimension doit être utilisée avant d'estimer les paramètres d'un classifier.
- **Sur-apprentissage** : Ce phénomène est l'un des plus importants lorsque l'on travaille dans le domaine de l'apprentissage.
- **Polysémie :** un mot ou une expression qui a plusieurs sens ou significations différentes comme le mot avocat qui peut désigner le fruit, le juriste et cela provoque une ambigüité.
- **L'homographie :** En linguistique, des mots homographes sont des mots qui s'écrivent de la même manière, tout en se prononçant ou non de façon différente et cela génère un bruit qui va causer une dégradation de précision.

6.2.La classification des textes Arabes

La langue arabe est l'une de six officielles langue de nations unies, c'est une langue très différentes des langues européennes, elle compte 28 consonnes et s'écrit et se lit droite à gauche. Les lettres changement de formes présentation selon leur position (au début, au milieu ou à la fin du mot), Um mot arabe s'écrit avec des consonnes et des voyelles .les voyelles sont ajoutées au-dessus ou au-dessous des lettres .elles sont nécessaire à lecture et la compréhension correcte d'un texte, elles permettent de différent des mots ayant la même représentation.

Le traitement automatique de la langue arabe présente des difficultés spécifiques. Nous présentons dans ce qui suit certaines de ces difficultés qui rendent le traitement automatique de la langue arabe difficile à réaliser.

- **La segmentation**

La segmentation c'est une phase non triviale pour tout application de TAL prétraitement d'un texte dans le but par la suit les traiter , le principe de segmentation est de segmenter un texte en unité de certain type (chaine de caractère en mots ou un mot en caractères ou un texte en paragraphes) commencer une analyse d'un texte sans le segmenter en phrases(ou mot ou caractères) conduit à des résultats peu fiables, de même avoir une mauvaise segmentation conduit à accumuler les erreurs du traitement automatique du texte.

Pour la langue arabe il y a peu de travaux sur la segmentation de texte et li n'existe pas des segmentations fonctionnelles et spécifiques à la langue arabe [60].
La segmentation est source d'ambiguïtés, vu que d'une part la ponctuation est rarement utilisée dans les textes arabes et d'autre part cette ponctuation, lorsqu'elle existe, n'est pas toujours déterminante pour guider la segmentation. De plus, certains mots outils peuvent marquer le début d'une nouvelle phrase, ce qui nécessite des analyses de surface afin de pouvoir segmenter le texte.

Exemple :

Entrée : ولد هذا العالم والباحث في مصر وحفظ القران في سن العاشرة من عمره

Dans cette phrase, la particule 'و' [w] joue le rôle de séparateur entre propositions et segmente l'énoncé en deux propositions, par contre dans la phrase suivante :

Sortie :

La même particule ' و ' [w] ne joue pas le rôle de séparateur entre propositions mais plutôt celui d'une conjonction de coordination entre les mots العالم[*AlEAlm*] (Savant) et الباحث [*AlbAHv]* (chercheur) et donc ne segmente pas la phrase [61] .

- **L'absence des voyelles**

Les voyelles sont des signes diacritiques placés au-dessus ou au-dessous des lettres, La plupart des documents arabes sont non voyelles. En effet, les voyelles ne sont utilisées que dans certains ouvrages scolaires pour débutants et dans le Coran, Un texte arabe non voyelle est fortement ambigu. [61], les textes non voyelles fait des problèmes lorsque le traitement automatique de la langue.

- **L'agglutination**

La langue arabe est contrairement défirent de les langues européen et latines, un mot arabe peut parfois correspondre à toute une phrase اتعلمونهاcorrespond en français à la phrase "Est-ce que vous…?". Cette caractéristique engendre des ambiguïtés morphologiques au cours de l'analyse.

En effet, il est parfois difficile de distinguer entre la conjonction de coordinations mot et un caractère de mot [60] par exemple dans le mot "وهم " nous ne pouvons pas savoir si "و " (waw) est une lettre faisant partie du mot comme le cas de « wahmon » (imagination) ou le cas s'il s'agit de la conjonction de coordination "و " suivie du pronom personnel " وهم " « wa hom » (et + ils).

- **Ambiguïté**

L'ambiguïté est considérée aujourd'hui comme principale pierre d'achoppement du traitement automatique de la langue à une époque où la mémoire de stockage et la puissance de traitement des ordinateurs ne constituent plus un frein au développement d'application informatique. L'ambiguïté manifeste sous différents formes et selon différents niveaux de traitement que ce soient : lexical morphologique, syntaxique et même sémantique [62].

Exemple : Par exemple le mot " فهم " peut représenter un nom « fahmon » فهم (compréhension) ou un verbe « fahima » (il a compris), ou encore un pronom personnel précédé d'une conjonction de coordination « fã hom » (alors ils).

6.3.La classification des textes ourdous

Le traitement du langage naturel est devenu assez mature dans les langues occidentales. Mais dans le cas des langues d'Asie du Sud comme l'ourdou, les ressources linguistiques de base telles que les corpus, Word Net, les dictionnaires, les jeux de balises et les outils associés ne sont toujours pas disponibles.

L'ourdou est une langue du Pakistan, né de l'alliance de plusieurs langues étrangères à savoir l'Anglais, l'Arabe, le Persan, e le Turque. Toutes ces langues complémentaires ont une structure morphologique complexe, ce qui fait de l'Ourdou, une langue morphologiquement riche. Il n'existe pas de systèmes efficaces de traitement de texte et catégorisation automatique de l'Ourdou. De ce fait, il existe un besoin pressant de disposer d'un système de traitement de texte complet pour cette langue afin de pouvoir gérer les exigences de prétraitement [63] .

- **Richesse morphologie**

En TAL, la morphologie joue un rôle important. La morphologie est l'étude de la structure des mots Gupta et al. 2015 [64]. Le traitement automatique de la langue pour ourdou est importante en raison de sa nature unique de sa morphologie mais est difficile à traiter, cette langue est nouveau dans le domaine de traitement naturel de cette décennie, la plupart des travaux effectués sur l'ourdou dans ce domaine sont liée à sa morphologie, son orthographe et son écriture, la morphologie de l'ourdou est très riche et complexe ce qui en fait une langue stimulante pour les tâches de traitement de langage naturel car est inclusion de mots empruntés à des langues comme l'hindi, l'arabe, le persan, etc. Pour travailler sur le traitement de la langue ourdou il faut être très clair sur sa morphologie et son système morphologiques. ce qui signifie qu'en ourdou, il est possible que pour un même mot, il existe plusieurs variantes., En raison de sa forme riche et de sa nature de réflexion intense, il a toujours été le choix préféré des écrivains de poésie et pour cette raison, il est aussi appelé le langage de la poésie Riad 2010, 2007; Naz et al. 2012 [65]. En ourdou, nous avons plusieurs ordres de mots pour des phrases de sens similaire. Par conséquent, Urdus est également appelé langage de commande de mots libre Riaz 2010. (Deux phrases représentent le même concept avec un ordre déférent).

- **Manque de normalisation**

Un problème important que confrontent les chercheures en ourdou est le manque de normalisation des règles d'écritures de la langue. Il n'existe toujours pas de normes pour l'écriture des textes en ourdou, ce problème de rédaction de texte en ourdou crée beaucoup de difficultés dans la symbolisation de texte et la modélisation du langage.

- **Tokenization du texte et langue Modélisation**

Le manque de normalisation dans la rédaction des textes crée beaucoup de des difficultés dans la symbolisation des textes et la modélisation du langage engendré des problèmes lors de l'exécution de tâches de prétraitement de texte comme la Tokenization et le stem. On a trois problèmes majeurs : les mots composés tels que un mot peut être composé de plusieurs tiges pour construire un mot significatif (le tableau 4 montrer les différents format de mot avec des exemples) aussi les mot qui contiennent des alphabets Participants et non-participants talque Participants définit comme certains alphabets sont liés à l'alphabet précédent (ب پ ت ٹ ث ج چ ح خ س ش ص ض ط ظ ع غ ف ك ق گ ل م ن ه ھ ی) et non-participants (ے ء و ۇ ژ ز ر ذ ڈ د آ ا) donc Il devient difficile de symboliser ce type de mots sur la base ou le facteur espace. Parfois un mot peut s'écrire de deux manières : première avec 2 tiges et une deuxième où les deux tiges sont attachées cela s'appelle les postes connectés. Par exemple : (کون سا) (kon-sa) et (کونسا) (konsa) →laquelle.

Tableau 4: format de mot en ourdou

Forme de mot	Mot en ourdou	Traduction de mot
Composé AB	چپکے سے	sneak
Composé A-o-B	نظم و نسق	discipline
Composé A-e-B	وقت رخصت	temps de départ
Composé A-al-B	بیت المقدس	Holly house

Il y a d'autres problèmes courants avec la langue arabe comme Homonymes, Homophones, Marques diacritiques et Absence de majuscule et minuscule qui pose un problème de désambiguïsation du sens des mots.

- **Stemmer ourdou**

Les chercheurs ont proposé des systèmes de stemmer de la langue ourdou à base de règles, Parmi ces travaux citent [66]:

- Akram et al. (2009) ont proposé un algorithme qui basé sur des règles et nommé Assas-Band c'est le premier stemmer ourdou. Il fonctionne en trois phases comme suit :

1- Le système maintien des listes d'exceptions : la liste d'exceptions de règles préfixes (PrREL), la liste d'exceptions globales de préfixes (PrREL), liste d'exceptions globales postfixes (PoGEL) et la liste d'ajout de caractères (ACL).

2- la suppression de préfixe : si un mot donné possède un préfixe et que ce préfixe n'est trouvé ni dans PrGEL ni dans PrREL, supprimez ce préfixe.

- Sion si le mot ne possède aucun préfixe, il renvoie le mot d'origine

Exemple : بأخلاق اخلاق

3-La suppression de suffixe : Si un mot contient suffixe et que ce suffixe ne se trouve pas dans les listes d'exceptions globales Postfixe (PoGEL) tronque le suffixe.

- Sinon si un mot donné ne contient aucun suffixe, il renvoie le mot d'origine

Exemple : زند زند کی

4- Dans la dernière phase, après l'extraction préfixe / Postfixe, la racine obtenue est recherchée dans les listes de caractères d'addition (ACL), si elle est trouvée, sa lettre correspondante est ajoutée à la fin du mot et renvoyée comme une tige. Si le mot n'est pas trouvé dans la liste, il est renvoyé sous forme de tige.

Exemple :زند زنده

- Khan et al. (2012) a proposé un léger poids Stemmer LWS (Alight weight Stemmer for Urdu language*)* pour la langue ourdou. À l'instar d'Akran, Qurant-ul-ain (2009), LWS tient également à jour des listes d'affectations et d'exceptions d'affectation.

1- Les listes gérées comprennent : la liste de mots vides (SWL), la liste d'exceptions globale de préfixe (GPEL) et la liste d'exceptions globale de suffixe (GSEL).

2- le mot est vérifié dans SWL : s'il est trouvé, ignorez le mot de recherche et passez au mot suivant.

- Sinon si le mot de requête n'est pas trouvé dans la liste des mots vides, passez à l'étape suivante.

3- le préfixe du mot recherché est vérifié en GPEL: s'il est trouvé, ignorez le préfixe et passez à l'étape suivante.

- sinon supprimez le préfixe selon la liste de préfixes et procéder à l'étape suivante.

4- le suffixe est vérifié dans la liste d'exceptions globale du suffixe (GSEL) :s'il est trouvé, Stemmer le renvoie comme racine.

- sinon tronque le suffixe par rapport à la liste des suffixes et passe à l'étape suivante.

5- Dans la quatrième et dernière étape, le mot de requête est vérifié dans la liste des caractères ajoutés ACL s'il est trouvé, puis son caractère correspondant est ajouté à la fin du mot dérivé, sinon il revient comme une tige.

- Stemmer présenté par Gupta et al. (2013) fonctionne de la même manière que celui présenté par Akram et al. (2009) mais ils n'incluent pas les liste d'exception Ils utilisent uniquement les listes préfixe qui contient 12 préfixe et suffixe qui contient 107 suffixe pour obtenir la racine du mot donné. Dans cet algorithme, un mot donné est vérifié dans les listes préfixes et/ou suffixe, si préfixes/suffixe ou les deux sont trouvés, alors il faut supprimer les affixes approprié et la racine est dérivée.

Tableau 5: Les avantages / Les inconvénients des algorithmes de Stemmer

Le non de chercheure	Le n on d'algorithme	Les avantages	Les inconvénients
Akram et al 2009	Assas-Band	-supprime les préfixes et les suffixes dans le cas applicable. -peut être utilisé dans la recherche d'information et TAL et l'exploitation de texte	-Exécution lente car de nombreuses listes sont utilisées. -grand espace requis pour stocker les listes. -Les mots composés ne sont pas traités.
Khan et al 2012	Alight weight stemmer for Urdu Language(LWS)	-Ce Stemmer supprime uniquement les préfixes et les suffixes. -Ce type de Stemmer donne de meilleurs résultats dans le domaine de la recherche d'informations	-Exécution lente car de nombreuses listes sont utilisées. -grand espace requis pour stocker les listes. -Les mots composés ne sont pas traités
Gupta et al. 2013	Rules based stemmer in Urdu	-supprimer les préfixes et suffixes. -L'exécution rapide et l'utilisation des listes. - Bon pour la demande de recherche d'informations. - simple pour l'implémentation	-Ne pas gérer les mots pluriels irréguliers. - Les mots composés ne sont pas gérés

6.4. La classification des textes Persans

La langue Persan est une langue indo-européenne parlée et écrite principalement en Iran, au Tadjikistan et dans certaines régions d'Afghanistan. L'alphabet farsi contient 32 lettres. Le persan s'écrit de droite à gauche. D'autres langues comme l'Arabe, le Kurde et l'Ourdou utilisent la forme d'écriture du Persan mais ont leurs propres spécificités. Le Persan a également ses propres spécifications et polymorphismes de l'écriture [67].

Le persan est une langue difficile dans le domaine de traitement automatique de la langue. L'orthographe de droite à gauche la morphologie complexe, les règles grammaticales complexes et les différentes formes de lettres en font un langage intéressant pour la recherche en TAL [68] :

- **Lettres importées de l'arabe**

Le persan compte 32 lettres dans son alphabet qui recouvrent 28 lettres arabes. En outre, il existe des sons importés d'arabe tels que "Tanwin" et "Hamza" que nous utilisons dans certains mots importés en Persan. Ces mots peuvent être écrits sous différentes formes. Par exemple, "مسأله", "مسئله" et "مساله" sont toutes des formes d'écriture du mot "problème".

- **Ambiguïté de l'Unicode**

Il existe des lettres telles que "ی" (i) et "ک" (k) pour lesquelles nous avons deux Unicode (un pour le persan et un pour l'arabe). Comme certaines applications utilisent le premier et d'autres le second, nous devons unifier leurs occurrences avant de traiter le texte.

- **Différents orthographes**

Certains mots peuvent être écrits avec des lettres différentes comme " بلیت"et " بلیط" pour "ticket".

- **Espacement différent**

En persan, l'espace n'est pas un signe de délimitation et de frontière déterministe. Il peut apparaître à l'intérieur d'un mot ou entre les mots. En revanche, il peut ne pas y avoir d'espace entre deux mots. Dans ces situations, le persan sera similaire à certaines langues asiatiques comme le chinois, sans espace entre les mots. Il existe de nombreux mots qui peuvent être écrits avec un espace, un espace court ou sans espace. Par exemple ' ,'می رفت ,'میرفت'' ,'می‌رفت' sont tous des formes de écriture de même mot.

- **Différentes prescriptions d'écriture**

L'APLL (2006) annonce des règles et la prescription pour écrire en persan. Malheureusement, ces règles varient d'une année à l'autre et comportent de nombreuses exceptions. Les systèmes de TAL peuvent donc recevoir des textes de styles différents.

Dans certains cas, reconnaître le style correct n'est pas une tâche facile. Les différentes prescriptions diffèrent dans le style d'écriture des mots, l'utilisation ou l'élimination des espaces à l'intérieur ou entre les mots, l'utilisation de diverses formes de caractères, etc.

- **Traductions**

L'écriture de mots étrangers (par exemple, l'Anglais) en persan peut entraîner certaines ambiguïtés dans le choix des lettres. D'autre part, comme ces mots ne se trouvent pas dans les lexiques, la symbolisation et la vérification de l'orthographe ne sont pas faciles.

- **Des mots nouveaux**

Le persan est une langue dérivée et générative dans laquelle de nombreux mots nouveaux peuvent être construits en concaténant des mots et des affixes. La possibilité de rencontrer un nouveau mot qui n'est pas disponible dans le lexique du système est donc élevée.

- **Les verbes irréguliers et composés**

En persan, les constructions des verbes sont le plus souvent irrégulières. De nombreux verbes composés peuvent être dérivés de noms et d'adjectifs et dans de nombreux cas, les parties de ces verbes ont des dépendances à longue distance.

- **Ezafe Construction**

La construction Ezafe est une construction spéciale en persan qui attache des noms à leurs modificateurs. Ezafe est une voyelle qui est prononcée mais pas écrite (dans la plupart des cas). L'ezafe non écrit pose généralement des problèmes de fragmentation et de traitement syntaxique et sémantique des phrases, tandis que les différentes formes d'écriture de l'ezafe créent des ambiguïtés au niveau de la symbolique et de l'enchaînement.

- **Stemmer persan**

Dans les langues naturelles, un mot typique est généralement composé d'une racine et quelque affixe. Dans la langue persan il n y a donc que des tiges et d'autres mots sont construits en ajoutant les préfixes et les suffixes.

Les Stemmer sont des logiciels qui permettent trouver les racines syntaxiques des mots, ils jouent un rôle important dans le traitement automatique de la langue naturelle et d'autres domaines tels que Système de Recherche d'Information SRI [69].

Un petit nombre de tige (Stemmer) ont été développées pour la langue persane, comme il n'est pas facile de formaliser les règles de grammaire et qu'il y a de nombreuse exceptions, il devient plus difficile de faire des Stemmer.

En 2002 le Stemmer *Bon* a été proposé par Tashakori et al cet est le premier algorithme à utiliser pour le Stemmer, il existe deux autres algorithmes de Stemmer en langue persan [70]:

Le Stemmer kazem Taghva a été proposé en 2005 par les chercheurs *Kazem* et Russel Beckley et Mohammed sabeh, de l'institut de science d'information de l'université de Las Vegas, celui-ci est semblable à l'algorithme de porter en anglais, qui se base sur la suppression des suffixes et de préfixes.

Le second algorithme est conçu par GholamReza Ghsen sani et Reza Hesamifard la seconde méthode se base sur les informations de bases de données, en d'autre termes, toutes les racines doivent être sauvegardées dans la base de données qui permet d'effectuer des recherches. Cette méthode pose quelques problèmes comme la mise à jour de la base de données et la vitesse de la Stemmer qui est également faible [71].

La mise en place d'un Stemmer pour la langue Persane est très difficile car la langue Persane a une morphologie complexe dont certains mots sont partagés avec la langue Arabe. De plus, il y a des exceptions. Certains affixes peuvent être enlevés sans problèmes mais la suppression d'autres affixes produit des résultats différents des mots d'origine.

7. Conclusion

L'Identification de la Langue est une tache de traitement automatique de la langue importante qui peut être modélisée comme un problème de classification qui se base sur l'apprentissage machine. Nous avons introduit dans ce chapitre la classification automatique et son processus et ses différentes étapes et les applications de CT, nous avons décrit les deux types de catégorisation et les problèmes majeurs de classification pour chaque langue. Dans le prochain chapitre nous présentons les différents algorithmes d'apprentissage utilisés pour l'identification de la langue par catégorisation de textes. De plus, nous avons fait une explication détaillée pour chaque Algorithme d'apprentissage utilisé dans notre système d'identification en présentant quelques avantages et inconvénient pour chaque algorithme.

Chapitre 03

Les techniques de l’Identification

CHAPITRE 3 : LES APPROCHES DE L'IDENTIFICATION

1. Introduction

Actuellement, plusieurs approches de catégorisation coexistent. Parmi les plus utilisées nous citons: le modèle probabiliste Naïve Bayes, ou les modèles vectoriels de Machine à Vecteur de Support, les k-plus-proches voisins, ainsi que les modèles d'arbres de décision Dans ce travail, nous allons utiliser un ensemble d'algorithmes variant entre les approches statistique et linguistique. Chacun d'entre eux a son propre principe ainsi que leurs avantages et inconvénients seront présentés, dans le but d'atteindre un degré maximal de précision et d'efficacité.

2. Les approches statistiques

2.1. Algorithme Machine à vecteurs de support (SVM)

Support Vector machine (SVM) ou séparateurs à vastes marges est une technique relativement récente qu'elle a été introduite en 1992 par Vladimir Vapnik, Bernhard Boser et Isabelle Guyon [72].elle aujourd'hui est reconnues comme l'une des meilleures méthodes d'apprentissage, ils restent considérés comme lents mais n'a pas besoin de beaucoup de données d'entraînement pour commencer à fournir des résultats précis. Ont été largement utilisés pour LID, avec un grand succès, le premiers à utiliser un SVM sont été Kim et Park en 2007 [73].

Il existe trois cas notamment le cas simple où les données sont linéairement séparables, suivis par le cas général où elles ne le sont pas nécessairement. Enfin, le concept le très important d'utilisation des SVM dite de marge souple.

- **Le cas simple (**linéairement séparables**)**

Les données sont dites linéairement séparables s'il existe un hyperplan tel que toutes les données appartenant à la classe 1 se retrouvent d'un côté de l'hyperplan alors que celles de la classe −1 se situent de l'autre côté. Plus formellement :

$w.x + b = 0$

Tel que :

$w.x + b > 0$ pour tout x appartenant à la classe 1

$w.x + b < 0$ pour tout x appartenant à la classe −1,

w.x + b = 0 Dans ce cas, il est possible de la laisser non classée, d'utiliser une autre règle ou de l'assigner aléatoirement à l'une des deux classes.

Avec w = $(w_1, \ldots, w_n) \in R_n$ le vecteur des coefficients de l'hyperplane

Et b ∈ R un scalaire appelé le biais

L'idée est de choisir le meilleur hyperplane une infinité d'hyperplanes qui peuvent servir a la séparation dont l'espace de séparation est une surface de décision appelée le marge de séparation définie comme la plus petite distance entre les exemples de chaque classe et la surface séparatrice.

- **Cas général Non linéairement séparables**

Repose sur l'utilisation d'une technique appelée noyau de séparation (kernel en Anglais) représente un espace de grand dimensions, la Figure 6 [74] montre à gauche le problème linéament séparable et à droite le cas non linéament séparable.

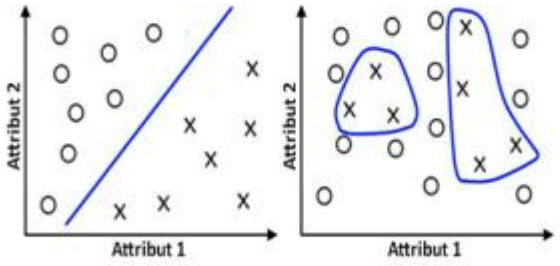


Figure 6: montre les différent cas linéaire et non linéaire

2.2. Algorithme de plus proche voisins (PPV)

L'algorithme des k plus proches voisins est connue en anglais sous le nom k-nearest Neighbors (KNN), est une méthode d'apprentissage supervisé, très connue dans le domaine de la catégorisation des textes, Son fonctionnement peut être assimilé à l'analogie suivante *"dis-moi qui sont tes voisins, je te dirais qui tu es…"* pour prédire où classer un nouveau document, il faut le comparer avec ceux déjà classés en cherchant ses ***K*** plus proches voisins. Une fois ces derniers déterminés, le nouveau document est classé dans la catégorie qui inclut le maximum de voisins parmi les ***K*** trouvés.

Deux paramètres sont utilisés : le nombre ***K*** et la fonction de similarité pour comparer le nouveau document à ceux déjà classés telle que la distance euclidienne par exemple qui est donnée par l'équation suivante :

$$d\,(x,y) = \sqrt{\sum_{j=1}^{m}\left(x_j - y_j\right)^2} \qquad (4)$$

Les K-PPV nécessitent :

- un ensemble de données D.
- Un entier **K**.
- une mesure de distance **d.**

2.2.1. Algorithme

Paramètre : le nombre K de voisins

Début

Pour chaque texte T faire

Transformer le texte T en vecteur T = (x_1, x_2, ..., x_m),

Déterminer les K plus proches textes du texte T selon une métrique de distance,

Combiner les classes de ces K exemples en une classe C.

Fin pour

Fin

Sortie : le texte T associé à la classe C.

La distance entre un texte et ses voisins se fait via une métrique de distance. Qui peut être :

- **Mesure Cosinus** qui consiste à calculer le produit scalaire entre deux vecteurs ***a*** et ***b***, le résultat sera divisé par le produit de la norme de ces deux vecteurs. La formule de la mesure Cosinus est alors la suivante :

$$Cosinus\,(a,b) = \frac{\sum(a*b)}{\sqrt{\sum a^2 * \sum b^2}} \qquad (5)$$

D’autres mesures ont été proposées dans la littérature, parmi lesquelles on peut citer les mesures de Jaccard et Dice.

- **Mesure de Jaccard** La formule de la mesure de Jaccard est alors la suivante :

$$J(a, b) = \frac{\sum(a*b)}{\sum a^2+\sum b^2-\sum ab} \qquad 6$$

- **Mesure de Dice** La formule de la mesure de Dice est alors la suivante :

$$D(a, b) = 2 * \frac{\sum(a*b)}{\sum(a^2+b^2)} \qquad 7$$

2.3. Arbre de décision (AD)

La construction des arbres de décision à partir de données est une discipline déjà ancienne d’où les premiers qui l’on utilisé sont Morgan et Sonquist (1963), depuis une vingtaine d’années des outils très populaires pour générer des règles de classification et plus généralement des règles de prédictions. concernant l’identification de la langue Häkkinen et Tian (2001) ont été les premiers utilisateurs des arbres de décision ("AD") en identifiant la langue d'un ensemble de données composé de quatre langues européennes, Les arbres de décision sont les plus populaires des méthodes d’apprentissage automatique, il sont utilisé pour résoudre des problèmes de décision séquentiels, sans aucune information de fréquence, permis les algorithmes d’arbre de décision les plus connus sont la Classification de document ID3 (Quinlan 1986) [75] consisterait à générer tous les arbres de décision possibles qui classent correctement l'ensemble de formation et à sélectionner le plus simple d'entre eux et C 4.5 (Quinlan 1993) [76].

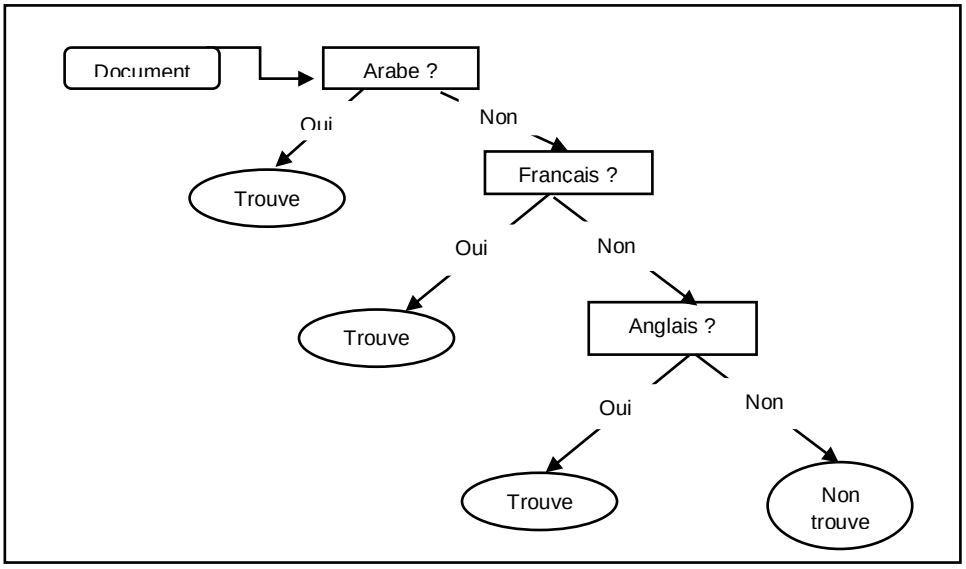


Figure 7: exemple de processus d'algorithme arbre de décision

2.3.1. Algorithme et le principe

Le principe des arbres de décision repose sur un partitionnement récursif des données. Le but du partitionnement est d'obtenir des groupes homogènes du point de vue de la variable à prédire. Le résultat est un enchaînement hiérarchique de règles.

La phase d'apprentissage consiste à la construction de l'arbre de décision avec sa racine, ses nœuds et toutes ses feuilles représentant l'ensemble des règles.

On a besoin de trois opérateurs permettant de :

- Décider si un nœud est terminal.
- Si un nœud n'est pas terminal, lui associer un test.
- Si un nœud est terminal, lui affecter une classe.

Algorithme générique :

```
Arbre ← arbre vide ; noeud_courant ← racine
Répéter
    Décider si le nœud courant est terminal
    Si le nœud est terminal alors lui affecter une classe
    Sinon sélectionner un test et créer autant de nœuds fils qu'il y a de réponses au test
    S'il existe un nœud Passé au suivant
```

Jusqu'à obtenir un arbre de décision consistant

Le principe de construction d'un arbre peut être alors décrit par les étapes suivantes

1. Calculer l'incertitude I(S) de la partition S.

2. Pour chaque attribut et chaque sommet candidats à la segmentation, calculer It(S) où St représente la partition issue de S après la segmentation d'un sommet selon l'attribut t.

3. Sélectionner l'attribut qui maximise la réduction d'incertitude $\Delta I(S) = I(S) - It(S)$ et effectuer la segmentation selon cet attribut.

4. S ← St

Le calcul du gain informationnel diffère d'une méthode à l'autre, mais le principe est le même.

Pour classer ou pour déterminer un nouveau document non étiqueté, il suffit de le soumettre à la racine de l'arbre et de le laisser parcourir les branches au fil des résultats des tests jusqu'à atteindre une feuille. Une série de questions qui ont été établies lors de la construction de l'arbre de décision seront posées. Les questions commenceront au nœud racine et s'arrêteront une fois qu'un nœud feuille sera atteint. Les réponses aux questions déterminent le cheminement de l'arbre qui sera suivi pour cet exemple. L'étiquette de classe qui se trouve dans le nœud de feuille atteint sera le résultat de la classification donnée par l'arbre de décision pour cet exemple [77].

2.4. La régression logistique(LR)

La régression

est une technique bien établie qui repose sur une base théorique solide, En 2006 Murthy et Kumar ont mis en œuvre un modelé régression linéaire multiple pour distinguer des paires été utilisée pour estimer le poids des caractéristiques et s'est avérée très efficace pour l'identification de la langue .Le modèle a été appliqué à l'identification par paires de langues parmi un ensemble des langues principale,c'est une technique statistique permettant d'étudier et de modéliser la relation entre les variables d'un système. Lorsqu'il y a plus de deux variables dans le système, on utilise le terme de régression multiple [78].

Bien qu'il s'agisse essentiellement d'une méthode de classification binaire, elle peut également être appliquée à des problèmes multi classes. Il existe deux principaux types de problèmes de classification :

- **Binaire ou classification binomiale**

(Régression linéaire) exactement deux classes à choisir (généralement 0 et 1, vrai et faux, ou positif et négatif).

- **Multi classe ou classification multinomiale**

(Régression logistique) c'est en fait une relation logarithmique entre trois classes ou plus de sorties à choisir.

À la différence de la régression linéaire (où la variable à expliquer est une variable quantitative), la régression logistique s'applique lorsque la variable à expliquer (Y) est qualitative.

Le but de la régression logistique est d'estimer les probabilités des événements ainsi que de déterminer une relation entre les caractéristiques et les probabilités de résultats particuliers.

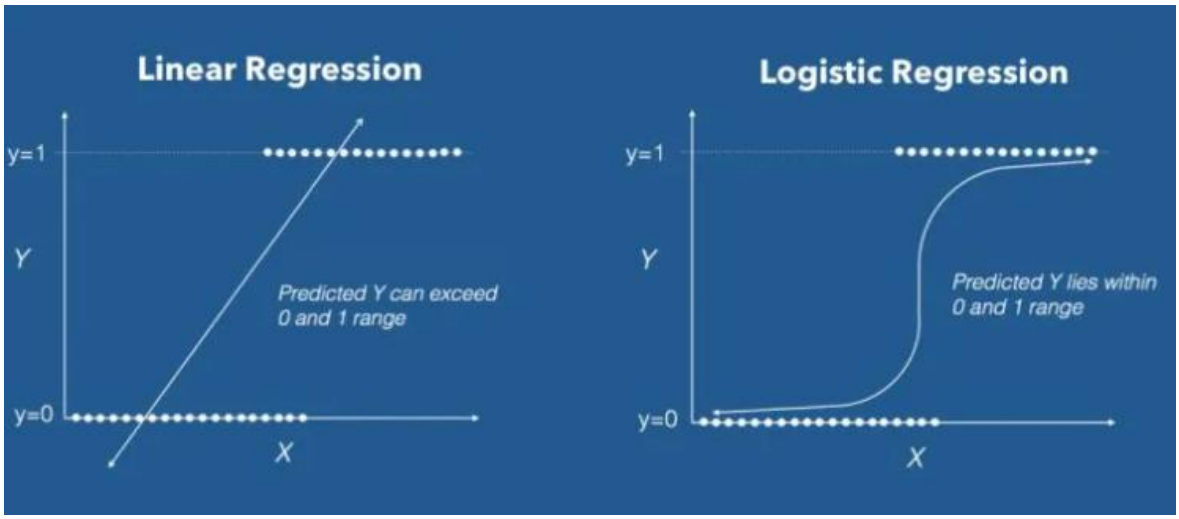


Figure 8: montre la Régression linéaire et la Régression logistique

2.5. naïve Bayes (NB)

Naïve Bayes est potentiellement bon pour servir de modèle de classification de documents en raison de sa simplicité.

La classification naïve bayésienne repose sur l'hypothèse que les attributs sont fortement (ou naïvement) indépendants. Elle est basée sur le théorème de Bayes qui ne s'applique que sous cette hypothèse. Théorème de Bayes est donné par :

$$P(x|y) = \frac{P(x:y)P(x)}{P(y)} \qquad 8$$

Avec $P(x|y)$= est la probabilité conditionnelle d'un événement x sachant qu'un autre événement y de probabilité non nulle s'est réalisé.

Le classificateur naïf de Bayes est l'exemple le plus simple de classificateur probabiliste il existe plusieurs modèle de classification de naïve bayes : Bernoulli naïve Bayes, Gaussian naïve Bayes, Bayes naïf multinomiaux, Complément naïve Bayes, Bayes naïve catégoriques

Dans notre expérience on a utilisé deux modèles qui :

2.5.1. Bernoulli naïve Bayes (BNB)

Modelé Bernoulli ou modelé d'indépendance binaire utilise des vecteurs d'occurrence de mot binaire (la valeur 1 si le mot correspondant est présent dans le document et 0 si le mot n'est pas présent), Bernoulli traite chaque message comme jetons E= $\{t_i,\ldots,t_n\}$ content une seule fois et la représentation par un vecteur binaire {0,1} et Alors le time élément de vecteur d'un document correspond au mot $w_t$dans le vocabulaire.

- Exemple

On a un vocabulaire V qui contient ensemble des mots

V= {Blue, Red, dog, cat, biscuit, Apple}

Et voilà le document =«the blue dog ate a blue cookie».

Si d est le vecteur caractéristique de Bernoulli pour ce document d = (1, 0, 1, 0, 1,0)

Pour classer un document, nous utilisons l'équation (7) Qui nécessite d'estimer les probabilités du document étant donné la classe P (D | C) et

Les probabilités a priori de classe P (C). Pour estimer la probabilité,

P (D | C), nous utilisons l'hypothèse Naïve Bayes appliquée a le modèle que nous utilisons.

Le modèle Bernoulli a la formule suivante :

$$p(d|c) = \prod_i^m (w_i p(w_i|c) + (1 - w_i)(1 - p(w_i|c))) \qquad (9)$$

M= est le nombre des mots, w_i est représenté la valeur de présence de i^e dans le document ou non.

P (w_i|c) est la probabilité de mot se produisant dans le document de la classe

(1-P (w_i|c)) est la probabilité de mot se produisant pas dans le document de la classe.

P ($w_i|c$) est estimée par :

$$\text{P } (w_i|c) = \frac{\sum_{j=1}^{n} w_{ij}\delta(c_j,c)+1}{\sum_j^n \delta(c_j,c)+2} \qquad (10)$$

Où n est le nombre de documents d'entrainement,c_j est la catégorie de j^e document, et w_{ji} est booléen indiquant la présence ou non du i^emot dans le j^e document d'entrainement.

2.5.2. Multinomial Naïve Bayes (MNB)

Un document est représenté par un vecteur d'entité avec des éléments entiers dont la valeur est la fréquence de ce mot dans le document.

- Exemple :

V = {Blue, Red, dog, cat, biscuit, apple}.

D= {2, 0, 1, 0,1 ,0} D est maintenant représenté par un Bag-Of-Word (BOW).

L'ordre des mots n'est ici pas considéré, mais bien la fréquence de chacun d'eux dans le texte. Dans ce modèle, la même hypothèse de NB est faite : la probabilité du nombre d'occurrences de chaque mot dans un document est indépendante de sa position et du nombre d'occurrences des autres mots du document. Estime la probabilité conditionnelle P (d |c) par :

$$P\,(d|c) = (\sum_{i=1}^{m} f_i)! \prod_{i=1}^{m} \frac{p(w_i|c)^{f_i}}{f_i!} \qquad 11$$

Où m= le nombre des mots.

w_i (i =1, 2,..., m) indique la présence i^e mot dans le document d.

f_i Est le nombre d'occurrences de $w_i$dans d.

P (w_i|c) est la probabilité conditionnelle que le mot wi apparaisse dans la catégorie

P (w_i|c) est calculé par la formule *suivante* :

$$P(w_i|c) = \frac{\sum_{j=1}^{n} f_{ji}\delta(c_j,c)+1}{\sum_{i=1}^{m}\sum_{j=1}^{n}\delta(c_j,c)+m} \qquad 12$$

2.6. La forêt aléatoire (RF)

À cause d'Instabilité qui représente un inconvénients principaux des méthodes d'apprentissage par arbres de décision donc la solution est d'apprendre plusieurs arbres sur l'échantillon d'apprentissage et de faire un vote sur les nouvelles données qui nomme Random Forest (RF) est un classificateur décrit comme un ensemble d'arbres de décisions entraînés introduites par Breiman, 2001 elles sont une combinaison entre les arbres de décision et la méthode d'ensachage qui propose par mm auteur en 1996 .

L'ensachage, une méthode permettant de générer un ensemble de données d'apprentissage par tirage aléatoire avec N exemples de remplacement, où N est la taille de l'ensemble d'apprentissage initial (Breiman, 1996), a été utilisé pour chaque caractéristique ou combinaison de caractéristiques sélectionnée.

Le classifier forêt aléatoire est bien approximé par le classificateur moyenné, Il a été utilisé avec succès en l'identification de la langue par Jhamtani et al. (2014).

2.6.1. Le principe et l'algorithme

Un ensemble S de n données

1) Créer K ensemble de données en réalisant un tirage avec remise

2) Apprendre K arbres de décision, un par ensemble de données

3) Retourner la forêt

Quand une nouvelle donnée doit être classée, on regarde la classe que donne chacun des K arbres : la forêt retourne la classe majoritaire (décision par vote majoritaire)

La clé du succès des forêts aléatoires est la façon dont elles créent chacun des arbres de décision qui composent la forêt. Il y a deux étapes de sélection aléatoire qui sont utilisées lorsque formant les arbres dans la forêt.

La première étape consiste à sélectionner au hasard, avec remplacement, les données des zones de formation fournies pour construire chaque arbre. Pour chaque arbre, un sous-ensemble différent de la sont utilisées pour développer le modèle d'arbre de décision et le tiers restant de Les données de formation sont utilisées pour tester l'exactitude du modèle. Les données de l'échantillon utilisées pour les tests sont souvent appelés les échantillons "hors sac".

La deuxième étape de l'échantillonnage aléatoire sert à déterminer les conditions de division pour chaque nœud de l'arbre. À chaque nœud de l'arbre, un sous-ensemble du prédicteur est choisi au hasard pour créer la règle binaire. Le nombre de variables prédicteurs qui sont sélectionnés de manière aléatoire peuvent être défini par l'utilisateur ou le choix peut être laissé à la forêt aléatoire algorithme. Utilisation d'un sous-ensemble de variables prédicteurs choisies au hasard pour diviser chaque nœud entraîne une moindre corrélation entre les arbres et, par conséquent, un taux d'erreur plus faible [79] .

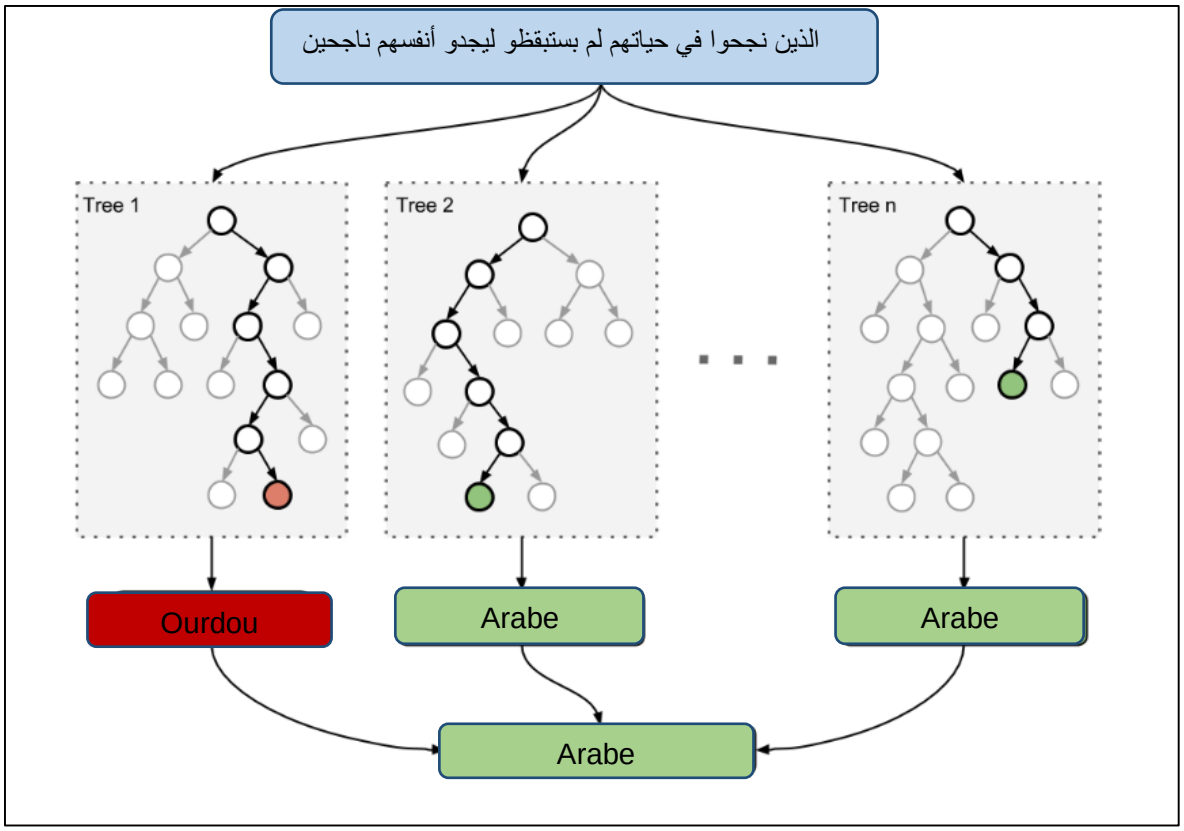


Figure 9 : exemple de processus d'algorithme Forêts aléatoires

2.7. Gradient Boosting (GB)

Contrairement à la forêt aléatoire, qui est un classificateur autonome et fusionné à l'aide de la moyenne de l'échantillon.

Gradient Boosting c'est un algorithme d'apprentissage automatique puisent qui peut être utilisé pour des problèmes de classification prédictive ou de modélisation de régression. Dérivés à l'origine par Friedman2001. L'idée principale du Boosting est d'ajouter de nouveaux modèles à l'ensemble de manière séquentielle. A chaque itération particulière, un nouveau modèle de base-apprenant faible est formé par rapport à l'erreur de l'ensemble de l'ensemble appris jusqu'à présent [80] .

Ilya plusieurs algorithmes de Boosting le Première c'est Adaptive Boosting (AdaBoost) et LightBoost, XGBoost.

2.7.1. Le principe du Boosting

Est considère comme un boite noire car c'est tu une méthode qui ne peut pas être explicité sous forme de règles de calcul.

Booster est une technique d'ensemble dans laquelle les prédicteur ne sont pas créés indépendamment mais il est basé sur une stratégie différente de manière séquentiellement, créé à partir d'arbres de décision ajoutés séquentiellement au modèle où les modèles suivants corrigent les performances des modèles précédents. Donc est une méthode générale pour améliorer la précision d'un algorithme d'apprentissage donné.

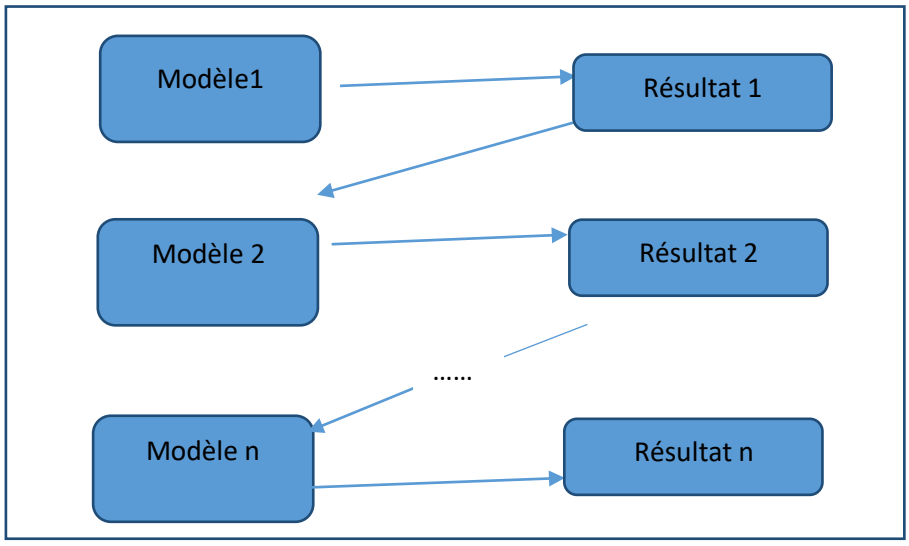


Figure 10: Processus de Gradient Boosting.

3. Les approches linguistiques

3.1. Approche base sur CT n-gramme

Est une catégorisation basée sur les fréquences n-gram des caractères talque n-gramme est une séquence de “n caractères “contiguë d'une chaîne de mot plus longue, Cette technique est une méthode de catégorisation des textes largement connue et fréquemment utilisée dans l'identification des langues, présentée dans Cavnar et Trenkle, 1994 [2].

L’idée est base à la représentation de document de donnée en n-gramme et utiliser le profil de fréquence des N-grammes pour classer les documents en fonction de leur langue talque base principalement sur a l'utilisation du n-gram pour l'identification des langues est que chaque langue contient ses propres n-grams uniques et tend à utiliser certains n-grams plus fréquemment que d'autres, fournissant ainsi un indice sur la langue. La figure 11 illustre l’idée générale de processus de cette approche.

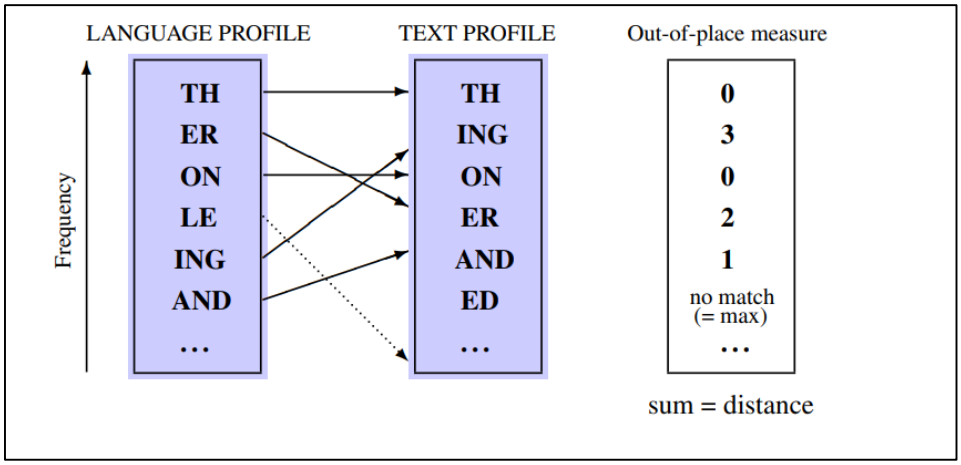


Figure 11: le processus de calcul de la mesure à distance.

3.1.1. Processus de catégorisation

- **Représentation de document en n-gramme** « profil n-grammes »

Apres élimination de caractère spacieux, pour faciliter la correspondance entre le début et la fin des mots nous représentons le blanc avec le trait de soulignement (“_“)si un n-gram comprend le premier caractère (par exemple, "_fi"), les caractères du milieu (par exemple, "dd", pas de soulignement ici), ou la fin (par exemple, nd_") d'un mot, c'est-à-

dire pour indiquer les limites du mot, génère tout le n-gramme possible et compte la fréquence de chaque un (rang). en utilisons un table de hachage et en fin trie le par ordre décroissant de leurs fréquences d'apparition dans le document.

- **Algorithme Mesurer la distance**

Pour calcule la similarité ces deux auteurs [Cavnar and Trenkle, 1994] proposent une méthode d'identification de la langue appelons Distance de Cavnar et Trenkle (CT).

Prend simplement deux N-gramme et calcule une simple statistique d'ordre de rang Cette mesure détermine la distance entre les N-gram.

Talque la distance égale rang de n-gramme en catégorie –rang de n-gramme en document

Plusieurs méthodes sont proposées pour mesurer cette similarité utilisés lors de la comparaison des profils (Méthodes de plus proche voisin, Distance de Beesley [1],…..).

La somme de toutes les valeurs déplacées pour tous les N-grammes est la mesure de la distance entre le document et la catégorie.et en fin choisi La langue diagnostiquée est celle pour laquelle la distance est la plus petite.

$$D_j = \sum_{i=1}^{N} |rank(t_i,\ text) - rank(t_i,\ l_j)| \qquad (13)$$

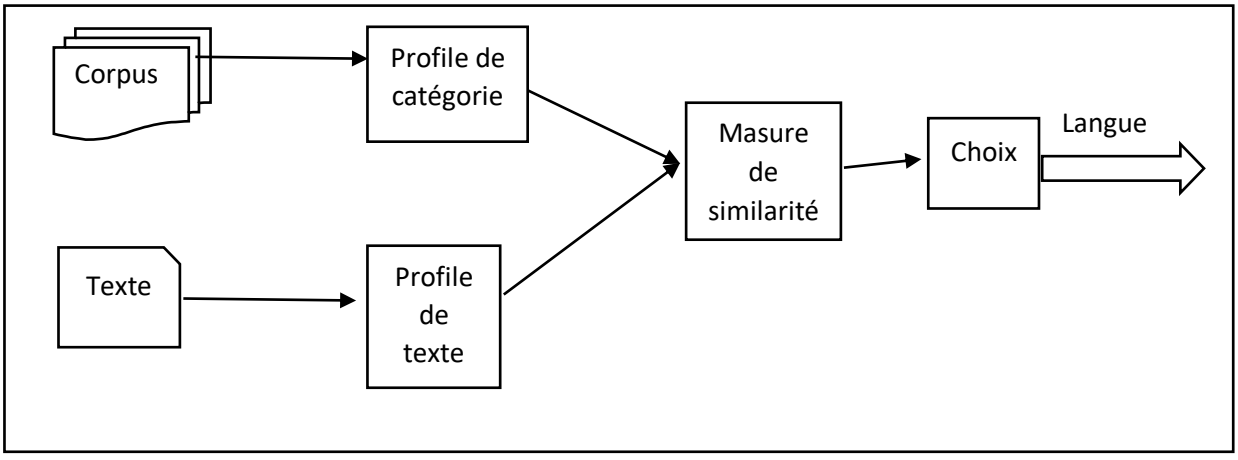


Figure 12: démarche de catégorisation par n-gramme

1.1.1. Trigrammes de lettres

Dans la littérature, le consensus s'est porté sur n=3, car n3 on génère beaucoup de colonnes. Les trigrammes de caractères semblent très efficaces et sont utilisés dans plusieurs applications.

L'identification de la langue est effectuée à l'aide La technique des vecteurs de fréquence trigrammes (Damashek, 1995) consiste à comparer une vecteur des fréquences de trigramme pour le texte à classer avec les vecteurs du langage dans un grand échantillon de langue, Ce tableau est montre que la technique de trigramme c'est la meilleur technique de n-gramme .

Le tableau 6 [81]représente la compression entre les expériences reportées par dans Gregory Grefenstette (1995) [8]qui compare la méthode de trigramme et les mots courts fréquents et les résultats de l'évaluation de n-gramme reportées dans JoanneCapstick (2000).

Tableau 6: compression entre n-gramme et trigramme et petit mot

	Method	3–5 words	6–10	11–15	16–20	21 or more
English	n-gram	68.75	97.1	99.3	99.7	100
	trigram	97.2	99.5	99.9	99.9	100
	short words	87.7	97.3	99.8	99.9	100
German	n-gram	95.1	98.4	99.2	99.9	100
	trigram	97.2	99.3	99.8	99.9	100
	short words	71.6	89.6	98.2	99.8	100
French	n-gram	84.9	98	98	98.3	99.5
	trigram	93	94.5	93.6	99.8	100
	short words	81.8	96	97.2	99.8	100

3.1.2. L'approche des N-grammes modifiés (MNG)

A été proposée par Choong et al.2009 [82] qui est une amélioration de l'approche originale des N-grammes, ne dépend pas de la fréquence des N-grammes, elle utilise plutôt une méthode booléenne pour décider du résultat de l'appariement. La méthode booléenne renvoie une valeur de 1 si les N-grammes des profils cibles se trouvent parmi les profils de formation. La méthode booléenne renvoie une valeur de 0 s'il n'y a pas de correspondance dans les profils cibles.

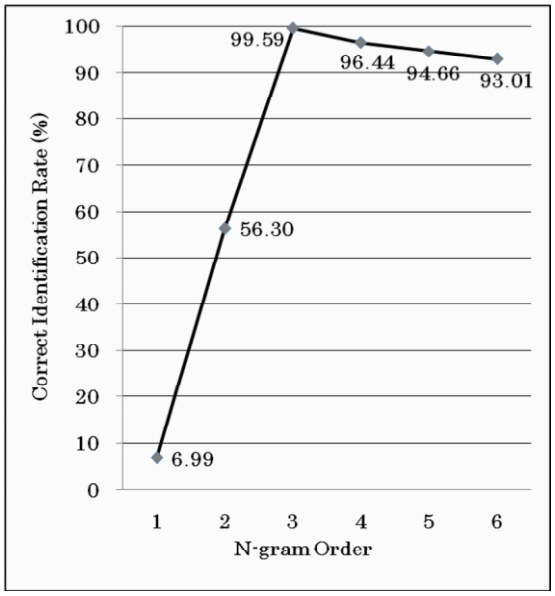
Une fois que tous les N-grammes des profils cibles ont été comparés aux profils d'apprentissage, le système calcule le taux de correspondance en divisant les valeurs

totales de correspondance et le nombre total de N-grammes distincts dans les profils cibles comme suit :

$$R_i = \sum_{i=1}^{n} \frac{m_i}{n} \qquad (14)$$

$$m_i = \begin{cases} 0, & si\ t_i ne\ correspondait\ pas\ \text{à}\ T_j \\ 1, & si\ t_i\ ne\ correspondait\ pas\ \text{à}\ T_j \end{cases}$$

Dans cet algorithme, l'unité de base du n-gram est du type de données "byte".



.

Figure 13: taux d'identification correct basé sur l'ordre de n grammes 1 à 6

3.2. La classification basée sur les centroïdes

La classification basée sur les Centroide est une approche d'apprentissage automatique utilisée dans le domaine de la classification des textes. Le site Le principal avantage des classificateurs basés sur les Centroide est leur haute performance tant au cours de la phase de l'apprentissage que l'étape du classement. Toutefois, le taux de réussite peut être inférieur à celui des autres classificateurs si un bon Centroide n'est pas utilisées. La classification basée sur les Centroide est très utile pour les tâches en ligne comme l'identification de la langue.

Est une méthode efficace basée sur le modèle de représentation vectorielle. Chaque document est représenté par un vecteur "D" et chaque élément du vecteur correspond à un terme de la collection de documents. On utilise généralement la métrique TF-IDF pour attribuer des poids aux termes.

La phase d'apprentissage consiste à calculer un centroïde pour représenter les classes, Le vecteur centroïde d'une catégorie sera représenté par la moyenne des vecteurs des textes qu'elle contient, qui correspond au barycentre des différents textes préalablement assignés à cette classe. Il existe principalement deux méthodes pour former les vecteurs centroïdes à partir des données de d'apprentissage (training) :

- **la moyenne arithmétique des Centroide** (arithmetic average centroid AAC), où les éléments d'un centroïde sont simplement les valeurs moyennes des poids des termes correspondants dans les vecteurs de documents appartenant à la classe. Le vecteur centroïde ~ci de la classe ci est formé comme suit :

$$\overrightarrow{c_i}=\frac{1}{|D_{c_i}|}\sum_{d\epsilon D_{c_i}}\vec{d} \qquad (15)$$

- est **le centroïde géométrique cumulé** (cumuli geometric centroid CGC) qui utilise la somme des poids des termes plutôt que leur moyenne :

$$\overrightarrow{c_i} = \sum_{d\in D_{c_i}}\overrightarrow{d} \qquad (16)$$

Il existe différentes variantes des méthodes de base de la CCA et de la CGC (la méthode CFC (class feature centroid) (Tan, 2008)).

La catégorisation d'un nouveau document se fait par comparer à chaque centroïde et il est attribué à la classe qui donne la valeur de similarité maximale.

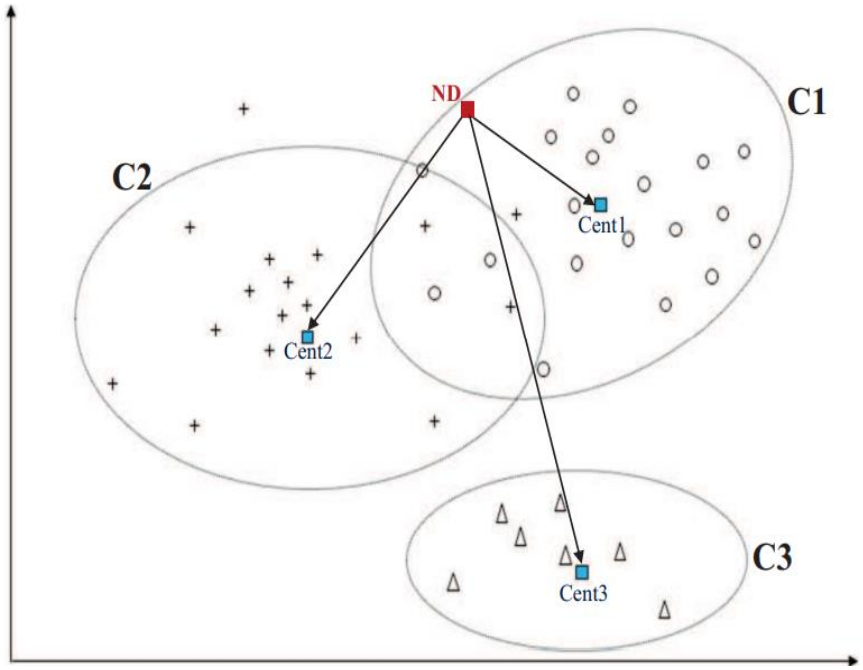


Figure 14: Exemple de la méthode du Centroide

Le nouveau document ND est attribué à la classe C1 de Centroide Cent1

3.2.1. Base sur Centroide de lettres

Des termes courts, des n-grammes ou des combinaisons de lettres uniques sont généralement utilisés dans l'identification de la langue. Toutefois, la taille des ensembles de caractéristiques de ces méthodes est importante et les coûts de prétraitement sont très élevés. Il est donc nécessaire de réduire leur dimension et les coûts de prétraitement, Les problèmes de dimension et de prétraitement peuvent en effet être résolus en utilisant des lettres dans l'identification de la langue [83] .

3.2.2. Méthode fréquence de classe inverse ICF (inverse class frequency)

La propriété la plus importante de l'ICF est son pouvoir de discrimination élevé pour les caractères spécifiques à la langue (caractères qui sont dans l'alphabet d'une ou de plusieurs langues seulement) Si l'on peut augmenter la l'importance de ces caractères lors de la classification, la performance sera plus élevée, puisque les classes seront davantage

séparées l'un l'autre, L'ensemble des caractéristiques comprend 54 éléments qui sont les suivants

suit:a,à,á,â,ã,ä,å,b,c,ç,d,e,è,é,ê,ë,f,g,g˘,h,ı,i,ì,í,ï,j,k,l,m,n,ñ,o,ò,ó,ô,õ,ö,p,q,r,s,s,t,u,ù,ú,û,ü,v, w,x,y,ß,z. Nous désignons les 26 lettres de l'alphabet anglais comme des lettres communes puisque chacune d'entre elles est utilisée dans une majorité des autres langues. Les autres lettres sont appelées lettres spécifiques à une langue.

Dans la CIF, nous formons d'abord les valeurs des Centroide en utilisant l'équation suivent

$$C_{ij}^{new} = log\left(\frac{\left(C_{ij}^{old} + \varepsilon\right)^{*}}{\sum_{i=1\ldots k} C_{ij}^{old}}\ sf\right) \qquad (17)$$

Où sf désigne un facteur de lissage de l'ensemble {1, 10,100, 1000,10 000} et ε< 0,001

Nous utilisons la fonction logarithmique et un lissage pour éviter de grandes déviations dans les valeurs des Centroide.

La figure 15 montre les déférences des répartitions des lettres entre les déférentes langues

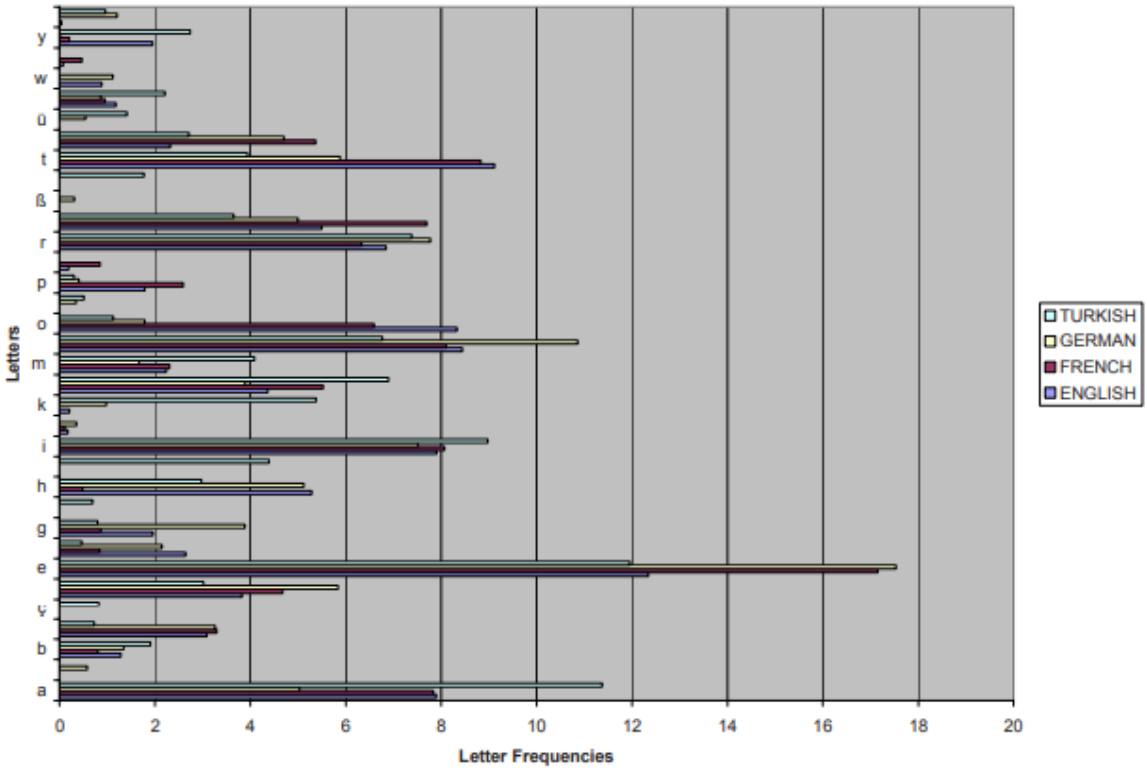


Figure 15: fréquence de lettre

3.2.3. Processus de méthode

Dans l'identification linguistique base sur les lettre les documents sont représenté par des distributions de lettres donc après la Division de texte en lettre séparés constitués uniquement de lettres et d'apostrophes. Les chiffres et la ponctuation sont éliminés, le système effectue les étapes suivantes :

1 -calcul la distribution des lettres de nouveau document brut D_{LF} ce distribution sont appelées "profils de document " chaque document est représenté sous forme de vecteur (de fréquence ou distribution dans un espace lettre.

D_{lf}= (lf1, lf2… lfin)

2 -les scores de texte de cette distribution de lettre sont multiples par les distributions de lettre moyenne de la langue Li.

$$score_i = D_{LF} * Li_{LF} \qquad 18$$

D_{LF} Distributions de lettres des nouveaux documents textuels

Li_{LF} Distributions de lettres moyennes des langues.

3 - après avoir calcul les scores de langue du nouveau document texte ils sont comparés les uns aux autres

4 - le score de langue maximum donne alors la classe du nouveau document

$$arg \underset{j = 1. \ \ldots, k}{max} \left(cos(x, C_j)\right) \qquad 19$$

Voice l'architecture de ce système, dans la figure 16.

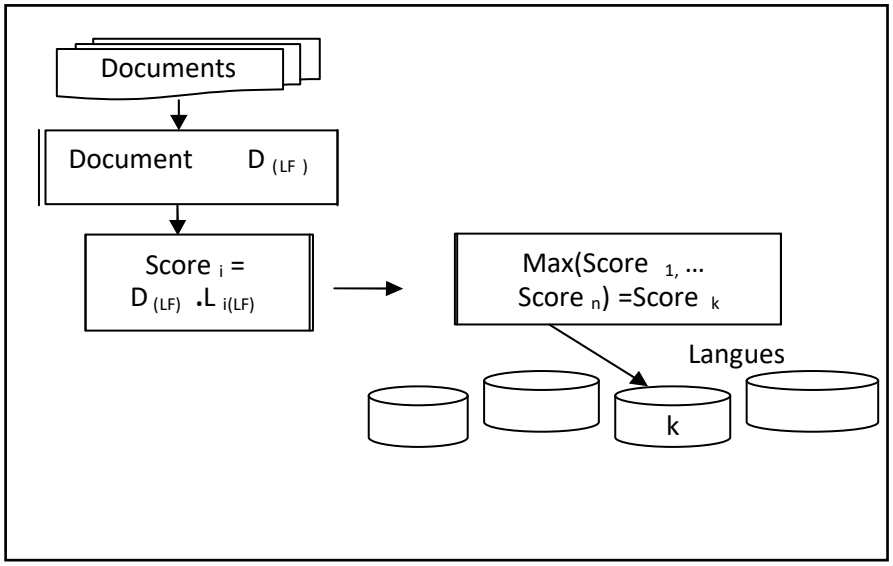


Figure 16 : Système d'indentification base sur les lettres

Algorithme

L'entraînement et les algorithmes de test pour la méthode ICF sont présentés dans la figure 17.Nous avons utilisé sf = 10 qui a donné les meilleurs résultats dans les expériences et e = 0,001 pour calculer les valeurs de l'ICF.

```
Module Train
Input
    Training set
Output
    Centroid vector c_i for each class c_i
Local variables
    freq_{i,k} : total frequency of letter l_k in class c_i
    freq_k : total frequency of letter l_k in the corpus
begin
    for each class c_i
        for each document d_j in class c_i
            for each letter l_k in document d_j
                freq_{i,k} = freq_{i,k} +1          // Increment class frequency of letter l_k
                freq_k = freq_k +1                  // Increment corpus frequency of letter l_k
            end for
        end for
    end for
    for each class c_i
        for each letter l_k
            c_ik = log((( freq_{i,k} +0.001)/ freq_k )*10)   // Calculate centroid value c_ik (sf=10)
        end for
    end for
end

Module Test
Input
    Test document d
    Centroid vector c_i for each class c_i
Output
    Class index of document d
Local variables
    f : frequency vector of letters in the corpus
begin
    for each letter l_k in document d
        f_k = f_k +1                                // Increment document frequency of letter l_k
    end for
    for each class c_i
        compute sim(f, c_i)
    end for
    return arg max_i sim(f, c_i)
end
```

Figure 17: algorithme base sur les lettres

Cas particulier : si tous les scores linguistique sont égaux à zéro ou si certain scores linguistique sont égaux les uns aux autres le classement ne peut pas être effectue.

4. Les avantages et les inconvénients

Chacun de ces modèles possède certains avantages et inconvénients que nous résumons dans le tableau 07 ci-après :

Tableau 7: Tableau des avantages et inconvénients des techniques

classifier	Avantages	inconvénients
plus proche voisins	- Simple à concevoir. - Flexible dans les cas de traitements non-linéaire séparables.	- Sensible aux bruits. - Très coûteux si le nombre de variables prédictives est très grand.
Arbre de décision	- Simple à comprendre et lisible. - Facilement interprétable. - Robuste au bruit : peut prendre en compte tous les type d'attribut. - Performent sur de grands jeux de données.	- Risque de surapprentissage. - La modification d'un seul nœud s'il est près du sommet modifie entièrement l'arbre.
Machine à vecteurs de support	- N'a pas besoin de beaucoup d'entraînement.	- Considéré comme lente.
Bayésien	- Très simple d'utilisation - Bon résultat lorsque les données est gros.	- Très sensible à leur corrélation.
Régression logistique	- Robuste et n'a pas de paramètres configuration de règles. - largement connu et compréhensible.	- Nécessite des échantillons de grande taille pour atteindre un bon niveau de stabilité.
Forêts aléatoires	- Plus rapide que « AD ». - Meilleurs résultats que « AD ». - Fonctionne efficacement sur de grandes bases de données. - Il dispose de méthodes pour équilibrer les erreurs dans l'ensemble de données non équilibrés.	- Risque de surapprentissage.
Centroïde	- Plutôt efficace (et rapide). - Robuste et interprétable.	- Difficile dans les cas de nombreuses classes.

Gradient bosting	- Adaptable aux spécificités des problèmes étudiés.	- Non explicite car est une méthode ensembliste.

5. Conclusion

Parmi les principale méthodes de l'identification de la langue les approche linguistique et statistique. Dans ce chapitre nous avons présenté notre choix des algorithmes dont nous aurons besoins dans notre travail. Nous avons par ailleurs, fait une petite synthèse en présentant les avantages et inconvénients de chaque algorithme. Dans le chapitre suivant nous présentons l'architecture de notre système, les corpus que nous avons utilisés pour nos tests.

CHAPITRE 4 :

Conception et modélisation de système

CHAPITRE 4 CONCEPTION ET MODELISATION DE SYSTEME

1. Introduction

Dans le présent chapitre nous présentons la conception de notre projet avec l'architecture de solution proposée, par la suite nous détaillons chaque étape de la solution proposée.

2. Objectifs

L'identification de la langue est une tache nécessaire qui prenait de plus en plus une importance, cette tache on peut la réaliser avec deux approches linguistiques et approche statistique chaque approche a des problèmes et des caractéristiques. L'objectif de notre système est de reconnaître la langue du texte parmi les six langues mentionnées ci-dessus , dans le but de faire des études et des expériences sur l'identification automatique de la langue et aussi d'étudier le problème d'identification de la langue pour les langues rapprochées (Arabe, Persan et Ourdou) de côté et (Anglais, Français et Allemand) de l'autre côté. Nous avons proposé une application web basée sur les mots comme unité d'identification.

3. Conception architecturale

Dans cette section, nous décrivons la méthodologie générale utilisée pour construire notre système, nous avons utilisé une solution hybride entre les approches statistiques (Machine Learning) et linguistiques. La figure 18 représente un schéma de l'architecture globale de la solution proposée pour l'Identification Automatique de la Langue .Notre solution est composée principalement des quatre étapes suivantes :

- Construction de corpus.
- Recherche de la configuration optimale.
- Etude de robustesse du système.
- Application.

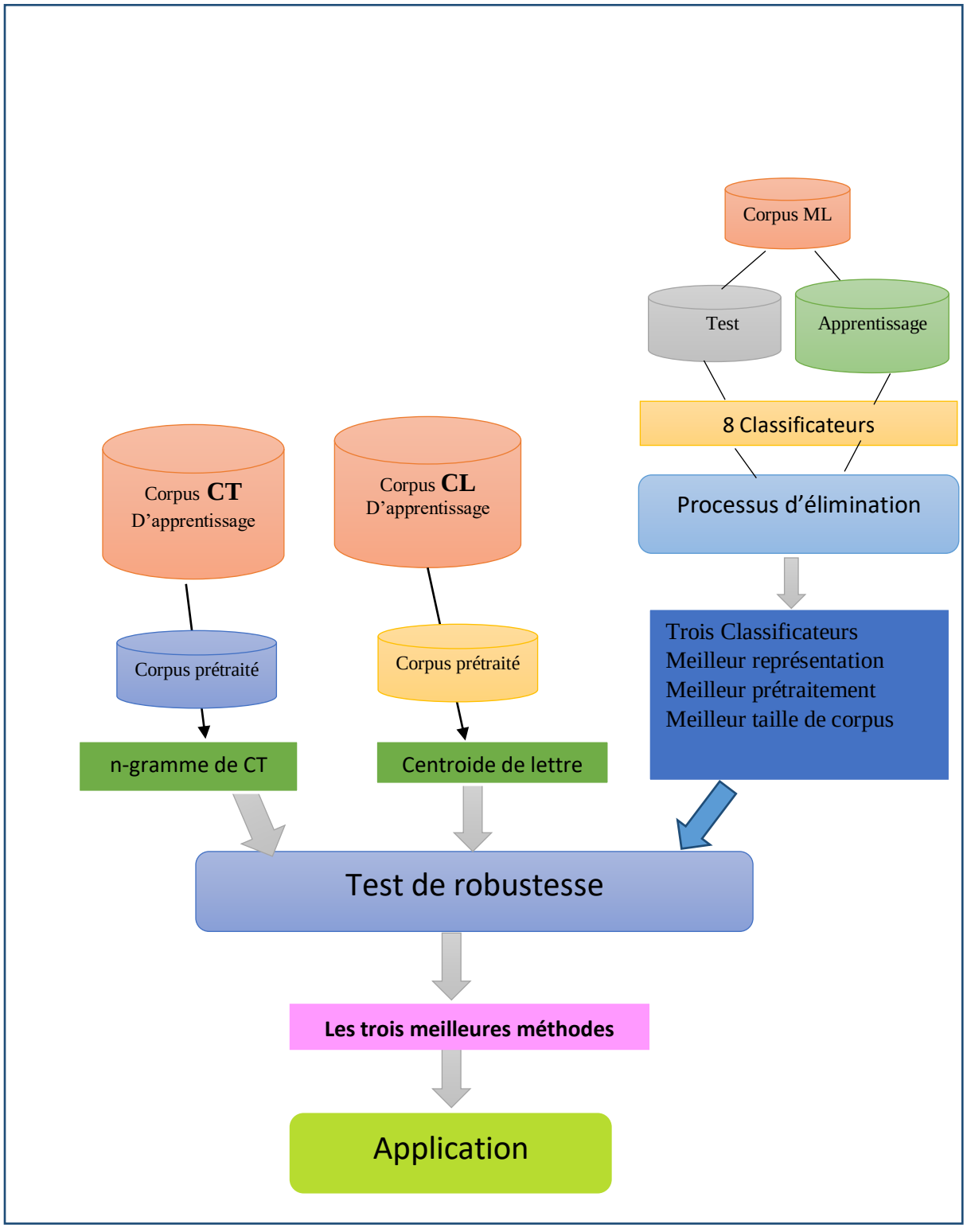


Figure 18: l'architecture globale

4. Analyse des besoins fonctionnels

La figure suivante représente le diagramme de cas d'utilisation de système d'identification, Dans l'application l'utilisateur est incapable d'identifier un texte ou un document, il n'est possible que de faire un test aléatoire. Pour identifier un texte ou un document il faut s'inscrire dans le site et remplir la fiche d'inscription et si la personne est déjà inscrite elle appuyé sur login et saisit le mot de passe et le nom d'utilisateur.

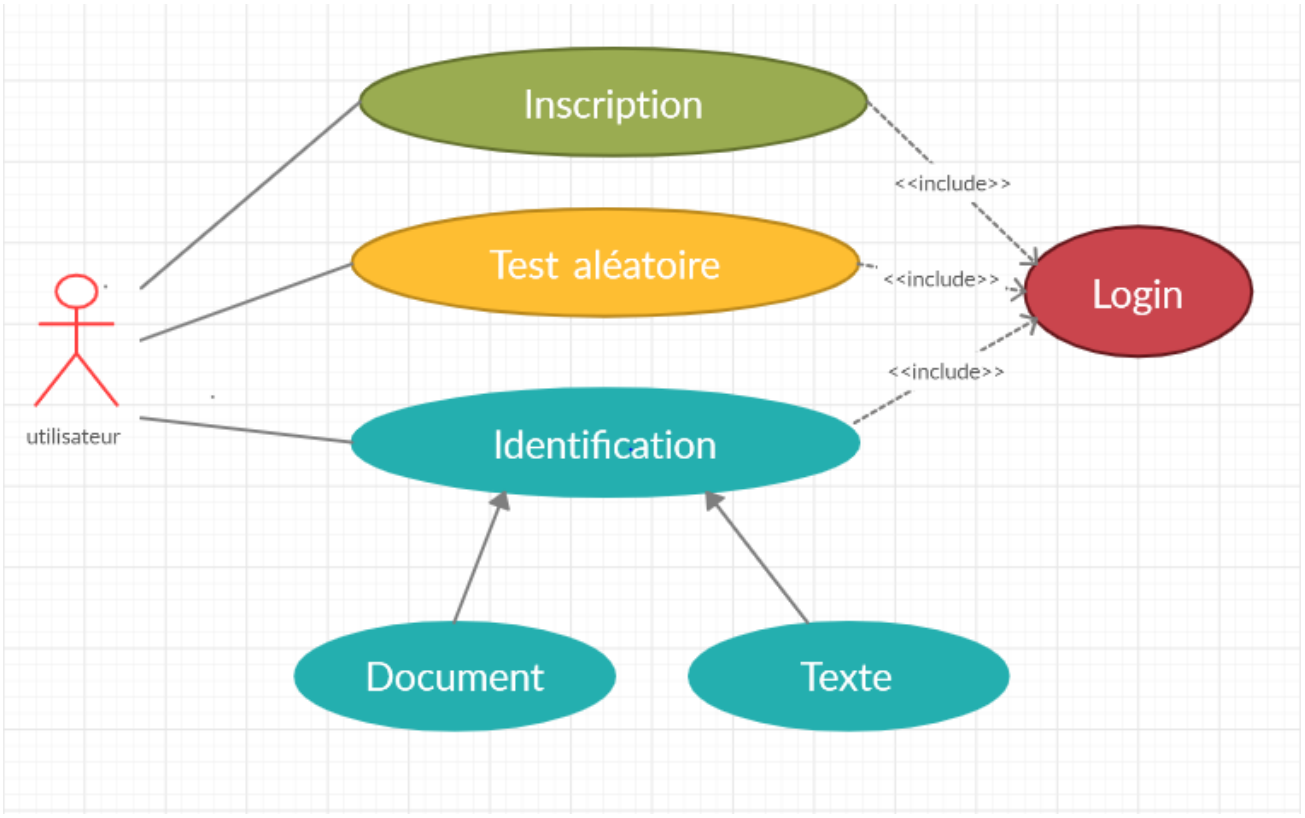


Figure 19: Digramme d'Activité de Système d'Identification

5. Conception détaille

Dans cette session nous détaillons notre conception et leur fonctionnalité.

5.1.Recherche de la configuration optimale

Nous utilisons une approche d'apprentissage supervisée avec huit classificateurs de machine Learning :Machine à vecteurs de support (SVM), plus proche voisins (KNN), Arbre de décision(AD), La régression logistique(LR), Bernoulli naïve Bayes(BNB), Multinomial Naïve Bayes(MNB), La forêt aléatoire(RF), Gradients Boosting(GB), et nous avons choisi deux type de prétraitement(élimination de mot vide ,suppression de ponctuations) et quatre représentations vectorielles (mot, caractère, n-gramme de caractère, combinaison) .

5.1.1. Prétraitement

Le prétraitement est la première étape pour l'Identification de la Langue. Dans cette section, nous expliquons quelques techniques et méthodes de prétraitement que nous avons faites sur les corpus d'apprentissage et les corpus de test.la figue 20 représentent le processus de prétraitement de notre solution tel que :

Le chemin en bleu : le prétraitement qui passe c'est la segmentation après la suppression de ponctuation.

Le chemin en noire : le prétraitement qui passe c'est la segmentation après la suppression des mots vides et à la fin la suppression de ponctuation.

Le chemin en jaune : le prétraitement qui passe c'est la segmentation après la suppression des mots vides.

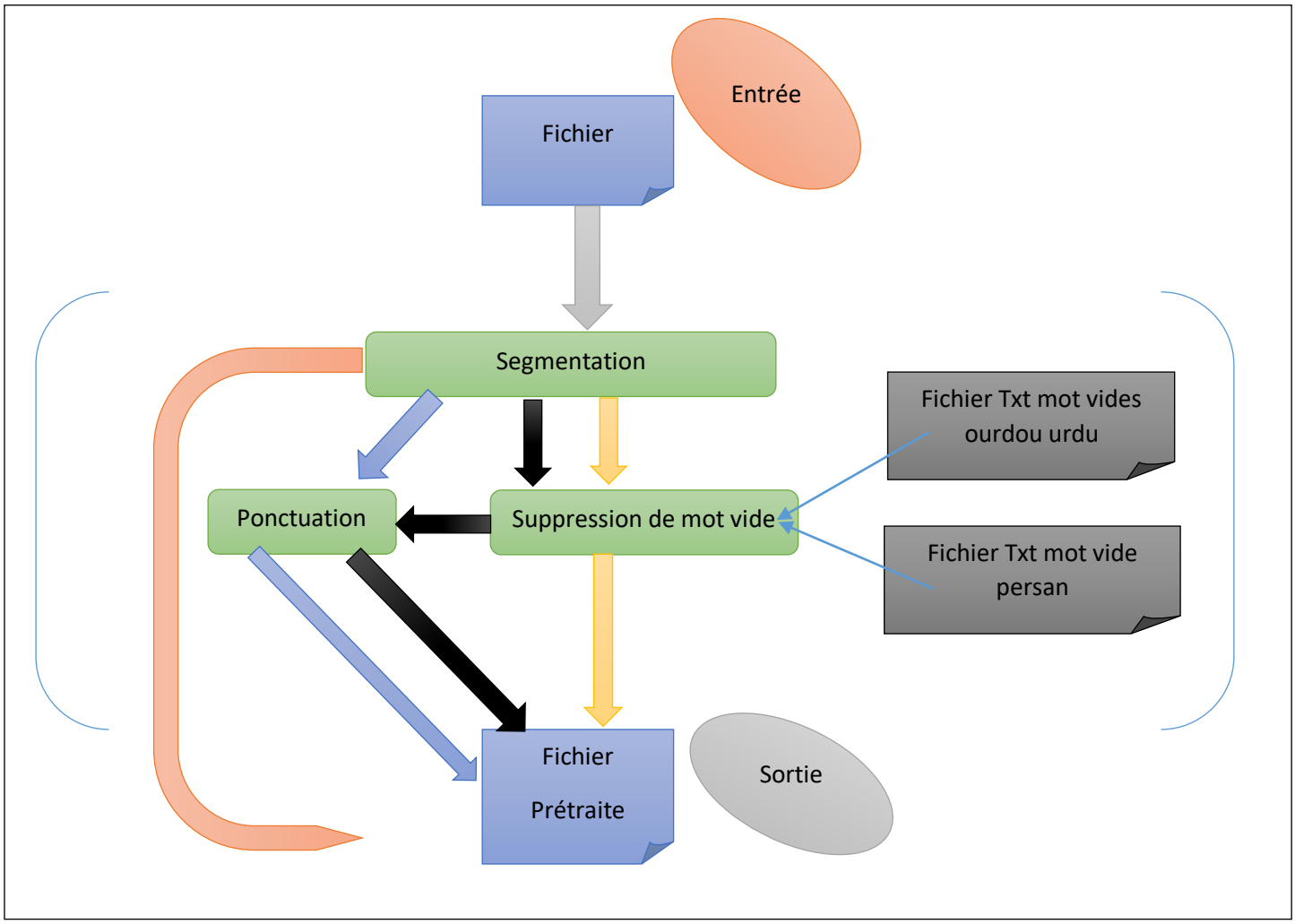


Figure 20: processus de prétraitement

- **Segmentation**

La segmentation est une méthode de prétraitement qui divise le texte, se produit à différents niveaux : un texte peut être divisé en paragraphes, phrases, mots, symboles ou phonèmes.

Exemple 1 :

Entrée : « كن انت المغير لا المدمر»

Sortie : « ‘المدمر’ , ‘لا’,’ المغير’,’ انت’,’ كن’»

Exemple 2 :

Entrée : « stay strong for yourself »

Sortie : « ‘stay’ , ‘strong’,’for ’,’ yourself’’»

Nous avons testé quatre différentes représentations vectorielles de textes (mot, caractère, n-gramme de caractère, combinaison)

Exemple : corpus test

- Au niveau de mot

Word_feat = ['corpus', 'corpus de', 'corpus de test', 'de' ,'de test', 'test']

- Au niveau de caractère

Char_feat= [' ',' d', ' de', ' de ', ' de t', ' t', ' te', ' tes', ' test', 'c', 'co', 'cor', 'corp', 'corpu', 'd', 'de', 'de ', 'de t','de te', 'e', 'e ', 'e t', 'e te', 'e tes', 'es', 'est', 'o', 'or','orp', 'orpu', 'orpus', 'p', 'pu', 'pus', 'pus ', 'pus d', 'r', 'rp','rpu', 'rpus', 'rpus ', 's', 's ', 's d', 's de', 's de ', 'st', 't','te', 'tes', 'test', 'u', 'us', 'us ', 'us d', 'us de']

- Au niveau de n-grammes de caractères

n-grammes de caractères d'un-gramme à cinq-gramme uniquement à partir du texte à l'intérieur des limites des mots, les n-gramme aux bords des mots sont remplis d'espaces (char-wb)

char_wb_feat = [' ',' c',' co',' cor',' corp',' d',' de',' de ', ' t', ' te', 'tes', ' test', 'c', 'co', 'cor', 'corp', 'corpu', 'd', 'de', 'de ', 'e', 'e ', 'es', 'est', 'est ','o', 'or', 'orp', 'orpu', 'orpus','p','pu', 'pus', 'pus ', 'r', 'rp', 'rpu', 'rpus', 'rpus ', 's', 's ', 'st', 'st ', 't', 't ', 'te', 'tes', 'test', 'test ', 'u', 'us', 'us ']

- Combinaison

combinaison tout ce qui précède car ces dernières années, on a eu tendance à tenter de combiner plusieurs types de caractéristiques différentes en un seul classificateur , sont un moyen de combiner différentes caractéristiques ou experts dans le but d'améliorer la précision grâce à une meilleure prise de décision.

all_feat= [' word_feat', 'char_feat', 'char_wb_feat']

- **Suppression de mots vides**

Est une technique qui permet de supprimer les prépositions, les mots de liaisons, les déterminants, les adverbes… qui représentent une grande part des mots d'un texte, mais malheureusement sont faiblement informatifs.

Exemple 1 :

Entrée : «la patience est la clé du salut »

Sortie : «'patience' , 'clé', 'salut'»

Exemple 2 :

Entrée : «do not give up the beginning is always the hardest »

Sortie : «'give','beginning','always','hardset' »

- **Ponctuation**

La ponctuation a pour but d'organiser le texte ou la phrase écrite grâce à un ensemble de signes graphiques On recense traditionnellement onze signes de ponctuation qui s'insèrent dans le texte : le point (.), la virgule (,), point-virgule(;),le point d'interrogation(?) ,les deux point(:)et les crochet([]) ...ect.

Exemple 1 :

Entrée : « vous sortez maintenant ? »

Sortie : « vous sortez maintenant »

Exemple 2 :

Entrée : «Mohamed : vous sortez maintenant !»

Sortie : «Mohamed vous sortez maintenant »

5.1.2. Présentation des expériences

Expérience : la détection de langue de différents typographes (écritures latins et écritures arabes). Nous avons testé tous les cas possibles où on a combiné les représentations vectorielles avec les prétraitements :

- Etape 0 :

Aucun prétraitement n'a été effectué, sauf les quatre représentations vectorielles (mot, caractère, n-gramme de caractère, tous) ce qui nous donne quatre expériences.

L'entrée de cette étape comporte les huit classificateurs et la sortie en comporte cinq, donc les trois restants seront éliminés par score.

- Etape 1 :

Un seul prétraitement a été effectué (soit l'élimination du mot vide ou la suppression de ponctuation) avec les quatre représentations vectorielles (mot, caractère, n-gramme de caractère, tous) et après la combinaison ça nous donne 8 expériences.

L'entrée de cette étape comporte les cinq classificateurs de la sortie de l'étape précédente, le score élimine deux classificateurs et donc la sortie en comporte trois.

- Etape 2 :

Les deux prétraitements appliqués à la fois avec les quatre représentations vectorielles sur les trois classificateurs issus de la sortie de l'étape précédente, nous donnent 4 expériences.

La figure 21 représente l'architecture détaillée de ce filtrage avec un seul corpus, nous avons conduit 16 expériences pour chaque corpus, et en répète ce processus avec tous les corpus et donc ça devient au total 48 expériences pour tous les corpus.

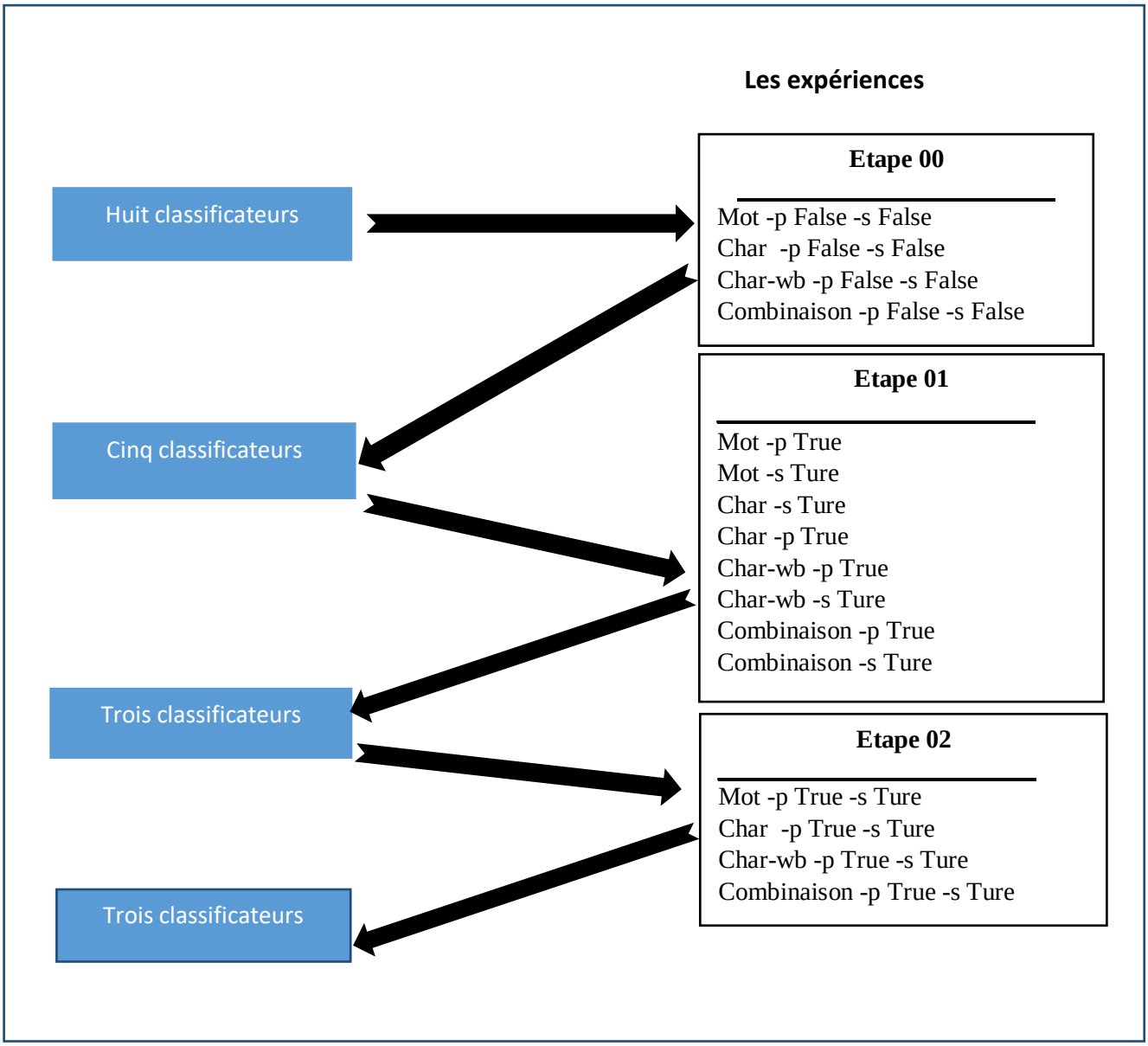


Figure 21: l'architecture détaille de processus d'élimination

À la fin de cette expérience nous avons réponde à ces questions :

- Quels sont les meilleurs classificateurs pour l'identification correcte de la langue ?
- Quelle est la représentation vectorielle optimale ?

- Quelle est l'importance de la présence ou de l'absence de ponctuation et du mot vide dans l'identification de la langue ?
- Quelle est l'efficacité du volume de corpus par rapport aux classificateurs ?

5.2.Etude de robustesse du système

Dans le but d'analyser le temps d'exécution et la sensibilité de la taille de fragment pendant l'identification correcte de la langue, nous avons testé la variation de scores et le temps d'exécution par rapport à la taille de fragment. Nous avons construit neuf corpus de tailles différentes (on commence avec la plus petite unité de mot avec 3 caractères et on augmente la taille afin d'observer les changements).

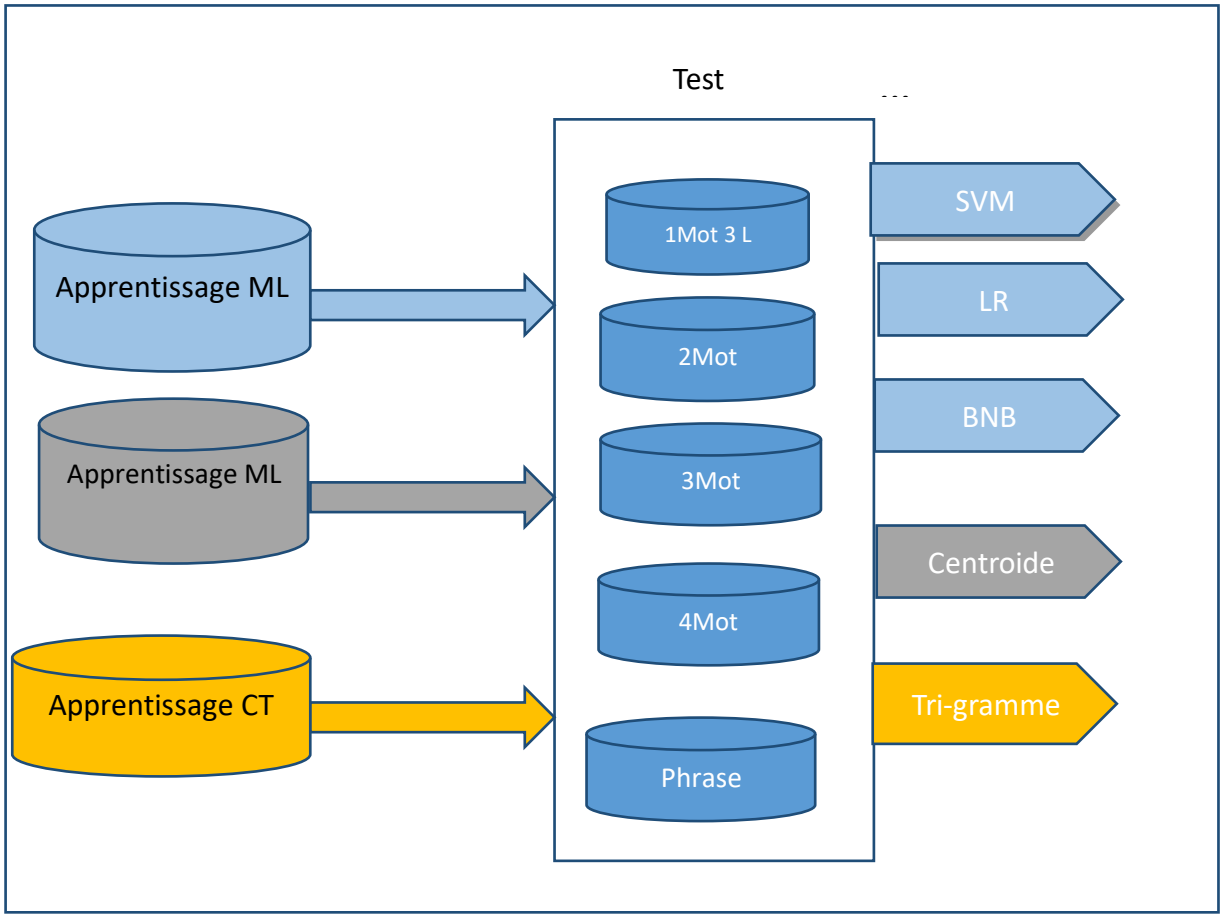


Figure 22: architecture détaillée de Test de robustesse

6. Conclusion

Dans ce chapitre nous avons présenté la conception de notre projet avec l'architecture de solution proposée, et nous avons détaillé chaque étape de la solution proposée.

Dans le chapitre qui suit, nous verrons comment nous avons fait toutes les expériences de chaque corpus et Comment nous avons fait les filtrages nécessaires pour les corpus et les représentations vectorielles et les méthodes d'apprentissage pour nous retenir à la fin à la meilleure représentation vectorielle et la meilleure méthode d'apprentissage, Enfin, nous utiliserons les résultats que nous avons obtenus pour développer notre application web.

Chapitre 05 : implémentation

CHAPITRE 5 : IMPLEMENTATION

1. Introduction

Apres avoir décrit notre solution de façon conceptuelle dans le chapitre 4.Nous exposons dans ce chapitre la description de l'environnement de développement et les Langages et les différents outils utilisés pour chaque étape de programmation. Ensuite nous montrons les résultats obtenus pour chaque partie. Enfin nous présentons le déroulement de notre application de l'identification utilisée de la langue.

2. L'environnement de développement

Le choix de l'environnement de programmation convenable est très important pour le développement des projets. Cela se fait suivant plusieurs facteurs : la facilité d'utilisation, la disponibilité de plusieurs fonctionnalités et plusieurs bibliothèques, la communication avec d'autres environnements… etc.

2.1. Environnement matériel

- Un ordinateur portable de type ACER :
  - ✓ Système d'exploitation : Windows 10 professionnel 32 bits.
  - ✓ Processeur : Intel® Celeron ® N2840.
  - ✓ Mémoire : 2GB.
- Un ordinateur portable de type DELL :
  - ✓ Système d'exploitation : Windows 10 professionnel 64 bits.
  - ✓ Processeur : Intel® Core ™ i5-5300U CPU @2.30 GHz.
  - ✓ Mémoire : 4GB.

2.2. L'environnement logiciel

Sublim texte

En 2007, Jon Skinner quitta son travail chez Google pour suivre de ses rêves, il a créé un meilleur éditeur de texte. Est un éditeur de texte générique codé en C++ Python et disponible sur Windows Mac et Linux, le logiciel est riche en fonctionnalités.

Sublime Texte prend en charge un certain nombre de langages de programmation différents : CSS, PHP, SQL, XML, PYTHON. ..Ect .La version actuelle de l'éditeur de texte Sublime est 3.0 et est compatible avec divers systèmes d'exploitation tels que Windows, Linux et Mac, OS.

Figure 23: Logo de sublime

- **Wamp**

WampServer [7]est une plateforme de développement Web, permettant de faire fonctionner localement (sans avoir à se connecter à un serveur externe). WampServer n'est pas en soi un logiciel, mais un environnement comprenant trois serveurs (Apache, MySQL et MariaDB), un interpréteur de script (PHP), ainsi que phpMYadmin pour l'administration Web des bases MySQL.

Figure 24 : Logo de Wamp serveur

- **Spyder**

Spyder[8] En 2008 Pierre Raybaut a créé et développé Spyder qui est un environnement scientifique puissant écrit en Python, pour Python. Spyder conçu par et pour des scientifiques, des ingénieurs et des analystes de données. Il présente une combinaison

[7] https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203603-sql-structured-query-language-definition-traduction-et-acteurs/
[8] https://docs.spyder-ide.org

unique des fonctionnalités avancées d'édition, d'analyse, de débogage et de profilage d'un outil de développement complet avec l'exploration de données, l'exécution interactive, l'inspection approfondie et les superbes capacités de visualisation d'un package scientifique. En outre, Spyder offre une intégration intégrée avec de nombreux packages scientifiques populaires, notamment NumPy, Pandas, Scipy, etc. En comparaison avec d'autres IDE pour le développement scientifique, Spyder a un ensemble unique de fonctionnalités - multiplateforme, open-source, écrit en Python et disponible sous une licence non-copyleft.

Figure 25 : Logo de spyder

- **Google Colab (Colaboratory)**

Est un service offert par Google (gratuit), basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement. Sans donc avoir besoin d'installer quoi que ce soit sur notre ordinateur à l'exception d'un navigateur.

Figure 26: logo de Google Colab

2.3. Langage de programmation et Bibliothèques

Pour implémenter notre travaille nous avons utilisé les outils suivants :

- **Python**

Le langage de programmation Python a été créé en 1989 par Guido van Rossum, aux Pays-Bas. La première version publique de ce langage a été publiée en 1991. La dernière version de Python est la version 3. Plus précisément, la version 3.8 a été publiée en octobre 2019.Ce langage de programmation présente de nombreuses caractéristiques intéressantes : -Il est multiplateforme(Windows, Mac OS, Linux, Android, iOS) C'est un langage de haut niveau, Le programme n'a pas besoin de compilé son programme pour pouvoir l'utiliser, Python est un langage interprété et il possède une riche bibliothèque et plusieurs fonctions qui faciliter le travail, Il existe une multitude de bibliothèques développées et fournies pour être utilisées en python .

Figure 27 : logo de python

- **Bibliothèque NLTK**

NLTK (Natural Language Toolkit) **:** est une plate-forme utilisée pour créer des programmes Python qui fonctionnent avec des données de langage humain pour une application dans le traitement du langage naturel .Cet ensemble a été créé à l'origine par Steven Bird et Edward Loper à l'Université de Pennsylvanie en 2001.[9]

Figure 28: logo de NLTK

[9] https://fr.accentsconagua.com/articles/code/introducing-the-natural-language-toolkit-nltk.html

- **Scikit Learn**

Scikit-learn est une bibliothèque de clés pour le langage de programmation Python qui est généralement utilisé dans les projets d'apprentissage automatique, Scikit-learn incluent les algorithmes de classification, de régression. [10]

Figure 29: logo de Scikit learn

- **Pandas**

Est un outil d'analyse de manipulation de données open source rapide puissant et facile à utiliser construit sur de langage de programmation Python.[11]

Figure 30: logo de Pandas

- **Pickle**

Est un module implémente des protocoles binaires pour la sérialisation et la désérialisation d'une structure d'objet Python[12].

Figure 31: logo de Pickle

[10] https://www.techopedia.com/definition/33860/scikit-learn
[11] https://pandas.pydata.org
[12] https://docs.python.org/3/library/pickle.html

- **RE**

Expression régulière, est une séquence de caractères qui forme un modèle de recherche. Re peut être utilisé pour vérifier si une chaîne contient le modèle de recherche spécifié.

- **Joblib**

Est un ensemble d'outils pour fournir un pipelining léger en Python est optimisé pour être rapide et robuste en particulier sur des données volumineuses

Figure 32: logo de Joblib

- **time**

La méthode de temps de Python time () renvoie l'heure sous forme de nombre à virgule flottante exprimé en secondes depuis l'époque,

- **Flask** [13]

Est un Framework Web créé initialement par Armin Ronacher comme étant un poisson d'avril. Le souhait de Ronacher était de réaliser un Framework web contenu dans un seul fichier python mais pouvant maintenir des applications très demandées. Flask vous fournit des outils, des bibliothèques et des technologies qui vous permettent de créer une application Web. Cette application Web peut être des pages Web, un blog, un wiki ou devenir aussi grande qu'une application de calendrier Web ou un site Web commercial.

En1^{er} Avril 2010, la première version a été publiée et la dernière version 1.1.2 est publiée le 30 Avril 2020.

Figure 33 : Logo de Flask

[13] https://pymbook.readthedocs.io/en/latest/flask.htm

- **Le langage SQL**[14]

SQL (Structured Query Language) est un langage informatique utilisé pour exploiter des bases de données. Il permet de façon générale la définition, la manipulation et le contrôle de sécurité de données. Dans la pratique, le langage SQL est utilisé pour créer des tables, ajouter des enregistrements sous forme de lignes, interroger une base de données, la mettre à jour, ou encore gérer les droits d'utilisateurs de cette base de données. Il est bien supporté par la très grande majorité des systèmes de gestion de base de données (SGBD). Créé au début des années 1970 par Donald D. Chamberlin et Raymond F. Boyce, le langage SQL est aujourd'hui reconnu comme une norme internationale, De nombreuses bases de données s'appuient sur le langage SQL

- **HTML**[15]

Signifie « *HyperText Markup Language* » qu'on peut traduire par « langage de balises pour l'hypertexte ». Il est utilisé afin de créer et de représenter le contenu d'une page web et sa structure. HTML fonctionne grâce à des « balises » qui sont insérées au sein d'un texte normal. Chacune de ces balises indique la signification de telle ou telle portion de texte dans le site. HTML permet d'inclure des images et d'autres contenus dans les pages web. Grâce à HTML, chacun peut créer des sites web aussi bien statiques que dynamiques.

Figure 34 : logo de HTML

- **CSS**[16]

Le terme **CSS** est l'acronyme anglais de *Cascading Style Sheets* qui peut se traduire par "feuilles de style en cascade". Le CSS est un langage informatique utilisé sur l'internet

[14] . https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203603-sql-structured-query-language-definition-traduction-et-acteurs/
[15] developer.mozilla.org/fr/docs/Web/HTML
[16] http://glossaire.infowebmaster.fr/css/

pour mettre en forme les fichiers HTML ou XML. Ainsi, les feuilles de style, aussi appelé les fichiers CSS, comprennent du code qui permet de gérer le design d'une page en HTML.

Figure 35 : Logo de CSS

- **Java script** [17]

en 1995 Brendan Eich a créé le Langage JavaScript en même temps que la technologie, ce Langage désigne un langage de développement informatique, et plus précisément un langage de script orienté objet. On le retrouve principalement dans les pages Internet. Il permet d'introduire sur une page web ou HTML des petites animations. Il existe de nombreux Framework JavaScript orientés vers les interfaces web, les trois plus connus sont JQuery, AngularJS (qui a été initialement développé par Google) et React (qui, lui, est né chez Facebook).

Figure 36: Logo JavaScript

3. Présentation des corpus

La catégorisation de textes est un processus long et complexe. Il est basé sur l'apprentissage dont la ressource primordiale est une base de données d'apprentissage nommée corpus.

[17] https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203585-javascript/

Un corpus (data set) est un ensemble de documents artistiques (textes, images, vidéos ... etc.) il est le premier type de ressources indispensables pour le développement. Le Web est aujourd'hui une source importante d'extraction de corpus.

3.1. Les corpus des méthodes Statistiques

Nous avons utilisé pour les algorithmes de machines Learning les corpus de textes suivants :

- 1er corpus (Language Identification data set)[18]

Ensemble de données d'identification de la langue de format csv. Ces données sont extraites de l'ensemble de données Wikipédia WiLi-2018 qui contient 235000 paragraphes de 235 langues. Chaque langue de cet ensemble de données contient 1 000 lignes.

Ce corpus contient 22 langues et nous utilisons les 6 langues sélectives de l'ensemble de données (arabe, persan, ourdou, allemand, français, anglais) au total 6000 phrases et 2 colonnes.

- 2eme corpus [19](TED Multilingual Parallel Corpus)

Les corpus parallèles TED : créé par M. Ajinkya kulkarni sont une collection croissante de corpus parallèles bilingues, de corpus parallèles multilingues et de corpus monolingues extraits des conférences TED[20] pour 109 langues du monde. Il comprend un corpus monolingue, 12 langues pour un corpus parallèle bilingue de plus de 120 millions de phrases alignées et 13 langues pour un corpus parallèle multilingue avec plus de 600 000 phrases de format Txt.

[18] https://www.kaggle.com/zarajamshaid/language-identification-datasst
[19] https://github.com/ajinkyakulkarni14/TED-Multilingual-Parallel-Corpus/issues
[20] www.ted.com

Tableau 8: le nombre de phrase pour chaque langue pour 2eme corpus

Langue	Le nombre phrase
arabe	553483
Ourdou	19861
persan	362411
allemand	471902
Français	493026

Par ailleurs, nous avons construit des corpus qui contiennent six langues (arabe, persan, ourdou, français, anglais, allemand) et nous avons commencé nos tests avec un petit volume avec le premier corpus dont nous avons progressivement augmenté la taille :

- Mini corpus : 1000 phrases pour chaque langue. Ce qui fait un total de 6000 phrases.
- Corpus moyen : 10000 phrases pour chacune des langues : arabe, persan, ourdou, français, allemand et pour l'Anglais 1000 phrases. Ce qui fait un total de 51000 phrases.
- Grand corpus : 20000 phrases pour chacune des langues : arabe, persan, ourdou, français, allemand et pour l'Anglais 1000 phrases. Ce qui fait un total de 101000 phrases.

3.2. Pour la méthode Linguistique

- La Leipzig Corpora Collection (LCC)[21]

(Biemann 2007) offre un accès à 195 langues où les données sont sous forme de fichiers texte (.TXT) de taille 10K ,30K, 100K ,300K, 1M phrases. Les sources sont soit des textes de journaux, soit des textes collectés au hasard sur Internet(Wikipédia) nous avons utilisé ce corpus pour la méthode de centroïde de lettres. Le tableau 10 présente la liste des langues avec leurs sources et la taille que nous avons utilisée pour construire notre corpus d'apprentissage.

[21] http://corpora.uni-leipzig

Tableau 9: la Taille et la source de chaque langue

Les langues	La source	La taille
Anglais	Wikipédia 2012	Un million de phrase
Arabe	Wikipédia 2016	Un million de phrase
Ourdou	Wikipédia 2016	Un million de phrase
Persan	Wikipédia 2019	Un million de phrase
Allemand	Wikipédia 2016	Un million de phrase
Français	Wikipédia 2010	Un million de phrase

- Crubadan

Est un lecteur de corpus NLTK pour les fichiers n-gramme fournis par le projet Crubadan. Il prend en charge plusieurs langues présentes 2228 langues contiennent la fréquence de mot et les digrammes des mots et trigramme de caractère, nous avons utilisé ce corpus pour la méthode de n-gramme CT.

4. Prétraitement

4.1. Suppression de ponctuation

```
def removePunctuation(text):
  out = text.str.replace('[^\w\s]','')
  return out
```

Figure 37: la fonction de suppression de mot vide

4.2. Suppression des mots vides

La bibliothèque NLTK supporte les mots vides de différentes langues comme Français, Anglais Arabe, Allemand mais ne supporte pas les langues ourdou et persan donc en fait une liste des mots vides pour ces deux langues en fichier de format Txt on code UTF-8 figure 38 et 39.

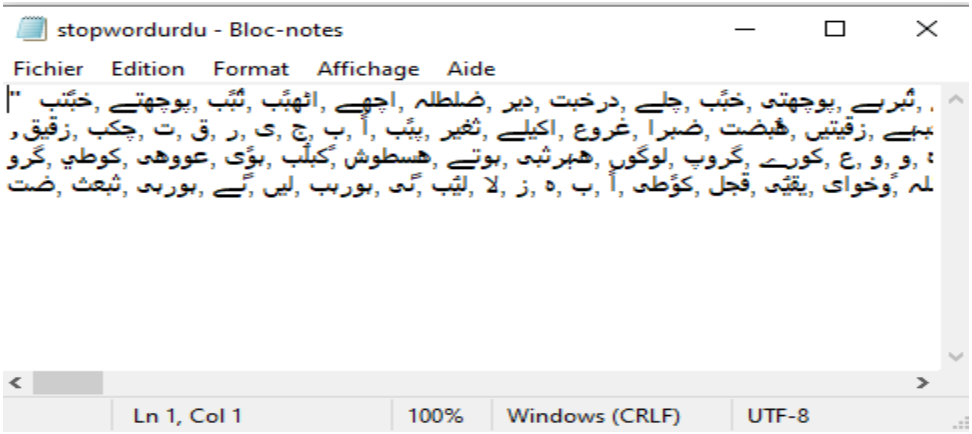


Figure 38: La liste des mots vides Ourdou

Figure 40: La liste des mots vides Persan

```
def removeStopWord(text):
  stopARB = stopwords.words("arabic")
  outARB = text.apply(lambda x: ' '.join([word for word in x.split() if word not in (stopARB)]))

  stopFR = stopwords.words("french")
  outFR = outARB.apply(lambda x: ' '.join([word for word in x.split() if word not in (stopFR)]))

  stopEN = stopwords.words("english")
  outEN = outFR.apply(lambda x: ' '.join([word for word in x.split() if word not in (stopEN)]))

  stopUR = open('stopwordurdu.txt','r', encoding='utf-8').read().split(',')
  outUR = outEN.apply(lambda x: ' '.join([word for word in x.split() if word not in (stopUR)]))

  stopPR = open('stopwordPr.txt','r', encoding='utf-8').read().split(',')
  outPR = outUR.apply(lambda x: ' '.join([word for word in x.split() if word not in (stopPR)]))

  stopGR = stopwords.words("dutch")
  outGR =outPR.apply(lambda x: ' '.join([word for word in x.split() if word not in (stopGR)]))

  return outGR
```

Figure 39: la fonction de suppression de mot vide

5. Expérimentation

5.1. Recherche de la configuration optimale

Dans cette session nous avons présenté les résultats détaillés pour chaque corpus et comme nous avons dit précédemment on a trois corpus de tailles différentes. Nous avons réalisé

```
!python textClassificationMultiling.py -tr corpusTrain51000.csv -trclm "Text, language" -
tst corpusTest51000.csv -tstclm "Text, language" -vec all > outExp115.log
```

Figure 41 : exemple de Shell script

notre teste avec Google Colab comme le montre la figure 41.

5.1.1. Résultat détaillé

5.1.1.1. Les résultats de mini corpus

- **Etape 00 :**

Nous avons remarqué dans cette étape que tous les résultats se situent entre 0,94 et 0,99 et le Meilleur classificateur est GB .GB a surpassé les autres classificateurs en atteignant un score des concurrents (SVM, LR) atteignent un score de 99,1 et 99,0% respectivement au niveau de caractère et n-gramme de caractère et tous mais KNN fonctionne très bien avec présentation par mot par rapport à l'autre représentation.

Le classificateur MNB fonctionne mieux que BNB et fonctionne très bien avec représentation par mot que les autres représentations.

Tableau 10: Résultats de l'Etape 00 1er corpus

Mot			Caractère			Caractère n-gram			Tous		
Class	model	Score	Class	model	Score	Class	Model	Score	Class	model	Score
1	MNB	0.990833	1	GB	0.994	1	GB	0,99333	1	GB	0.994167
2	KNN	0.990000	2	BNB	0992500	2	SVM	0,991667	2	LR	0.991667
3	RF	0.987	3	SVM	0,991	2	RF	0,991667	2	RF	0.991667
4	SVM	0.989	4	MNB	0990833	3	LR	0.990	2	SVM	0.991667
4	LR	0.989	4	LR	0.990833	3	MNB	0.990	3	KNN	0.9908
5	GB	0.9883	4	RF	0.990833	3	BNB	0.990	3	MNB	0.9908
6	DT	0.973	5	KNN	0.988	4	KNN	0,989167	3	BNB	0.9908
7	BNB	0.94	6	DT	0.985	5	DT	0,980833	4	DT	0.98

- **Etape 01**

Cette étape donne presque le même classement que celui de l'étape précédente mais nous remarquons que le résultat obtenu après la suppression de ponctuations est mieux qu'après la suppression de mot vide.

Tableau 11: Résultats de l'Etape 01 1er corpus

Suppression de ponctuation											
Mot			Caractère			Caractère n-gram			Tous		
Class	Model	Score	Class	Model	Score	Class	Model	Score	Class	Model	Score
1	MNB	0.990833	1	GB	0.99333	1	GB	0,993333	1	GB	0.995000
2	LR	0.989167	2	RF	0.991667	2	SVM	0,991667	2	RF	0.992500
2	SVM	0.989167	2	SVM	0.991667	3	LR	0,990833	3	LR	0.991667
3	GB	0.988333	3	MNB	0990833	4	MNB	0.988333	3	SVM	0.991667
4	RF	0.987500	3	LR	0.990833	4	RF	0.988333	4	MNB	0.990833
Suppression des Mots Vide											
Mot			Caractère			Caractère n-gram			Tous		
Class	Model	Score	Class	Model	Score	Class	Model	Score	Class	Model	Score
1	MNB	0.990833	1	GB	0.994167	1	GB	0,992500	1	GB	0.99333
2	LR	0.989167	2	RF	0.991667	2	RF	0,991667	2	SVM	0.991667
2	SVM	0.989167	2	SVM	0.991667	2	SVM	0,991667	2	LR	0.991667
3	GB	0.988333	3	MNB	0990833	3	MNB	0.988333	3	MNB	0.990833
4	RF	0.987500	3	LR	0.990833	3	RF	0.988333	4	RF	0.990000

- **Etape 02**

Les classificateurs discriminants (LR, SVM) obtiennent généralement de meilleurs résultats dans ces niveaux, nous avons calculé la moyenne de score de chaque représentation pour obtenir la meilleure représentation vectorielle.

La meilleure représentation est la combinaison de : mot, caractère et n-gramme caractère.

Tableau 12: Résultats de l'Etape 02 1er corpus

Suppression ponctuation et Mot Vide											
Mot			Caractère			Caractère n-gram			Tous		
Class	model	Score	Class	model	Score	Class	Model	Score	Class	model	Score
1	GB	0.987500	1	GB	0.994167	1	GB	0,99250	1	GB	0.99500
2	SVM	0.989167	2	SVM	0.991667	2	SVM	0,991667	2	SVM	0.991667
2	LR	0.989167	3	LR	0,990833	3	LR	0,990833	2	LR	0.991667
0.988611			0.992222			0.991666			0.992778		

5.1.1.2. Les résultats corpus moyen

- **Etape 00 :**

Le classificateur GB qui était en premier rang avec 99% à 98% dans le premier corpus, est devenu presque cinquième avec 98% à 93%, par contre on voit une amélioration des résultats du classificateur LR. DT donne toujours le même score même si le corpus devient plus grand. Le score de KNN a reculé vers 57% après ce qu'elle était à 99% et BNB a amélioré ses résultats par rapport à MNB qui a donné des résultats plus faibles avec Caractère, n-gramme caractère et combinaison mais ça a bien fonctionné avec les mots.

Tableau 13: Résultats de l'Etape 00 2émé corpus

Mot			Caractère			Caractère n-gram			Tous		
Class	model	Score	Class	model	Score	Class	Model	Score	Class	model	Score
1	MNB	0.991569	1	SVM	0.996569	1	BNB	0.997157	1	SVM	0.997745
2	SVM	0.990098	2	LR	0.995196	2	SVM	0.997157	2	LR	0.996176
3	LR	0.983529	3	BNB	0.995098	3	LR	0.995490	3	RF	0.994608
RF	RF	0.963333	4	RF	0.994020	4	RF	0.995098	4	BNB	.993922
4	DT	0.947059	5	KNN	0.989118	5	GB	0.990000	5	GB	0.990490
5	GB	0.939412	6	GB	0.987843	6	KNN	0.986863	6	KNN	0.987745
6	BNB	0.935098	7	DT	0.985098	7	DT	0.986176	7	DT	0.985098
7	KNN	0.571863	8	MNB	0.976765	8	MNB	0.976569	8	MNB	0.983137

- **Etape 01 :**

Si on compare les résultats de Suppression de ponctuation et Suppression des Mot Vide on trouve que ce sont très similaires.

Tableau 14: Résultats de l'Etape 01 2émé corpus

Suppression de ponctuation											
Mot			Caractère			Caractère n-gram			Tous		
Class	Model	Score	Class	Model	Score	Class	Model	Score	Class	Model	Score
1	SVM	0.990098	1	SVM	0.996569	1	SVM	0.997157	1	SVM	0.997745
2	LR	0.983529	2	LR	0.995196	2	BNB	0.997157	2	LR	0.996176
3	RF	0.964608	3	BNB	0.995098	3	LR	0.995490	3	RF	0.994118
4	GB	0.939510	4	RF	0.993529	4	RF	0.995196	3	BNB	0.993922
5	*BNB*	0.935098	5	GB	0.988039	5	GB	0.990196	4	GB	0.990490
Suppression des Mots Vides											
Mot			Caractère			Caractère n-gram			Tous		
Class	Model	Score	Class	Model	Score	Class	Model	Score	Class	Model	Score
1	SVM	0.997157	1	SVM	0.996569	1	SVM	0.997157	1	SVM	0.997745
2	BNB	0.997157	2	LR	0.995196	2	BNB	0.997157	2	LR	0.996176
3	LR	0.995490	3	BNB	0.995098	3	LR	0.995490	2	RF	0.994412
4	RF	0.994902	4	RF	0.993431	4	RF	0.994902	3	BNB	0.993922
5	GB	0.990098	5	GB	0.987843	5	GB	0.990098	4	GB	0.990392

- **Etape 02 :**

SVM et LR donnent des résultats Très proches et la meilleure représentation est la combinaison des trois : mot, caractère et n-gramme caractère.

Tableau 15: Résultats de l'Etape 02 2émé corpus

Suppression de ponctuation et Mots Vides											
Mot			Caractère			Caractère n-gram			Tous		
Class	model	Score	Class	model	Score	Class	model	Score	Class	model	Score
1	SVM	0.990098	1	SVM	0.999804	1	SVM	0.997157	1	SVM	0.999804
2	LR	0.983529	1	LR	0.998824	1	BNB	0.997157	1	LR	0.999804
3	BNB	0.935098	2	BNB	.993529	2	LR	0.995490	2	BNB	0.992843
0.969575			0.997385			0.996601			0.997483		

5.1.1.3. Les résultats Grand corpus

- **Etape 00**

Si on compare GB avec le corpus précédent on trouve qu'il a gardé le même rythme.

KNN a continué à baisser avec 26%.

Tableau 16: Résultats de l'Etape 00 3émé corpus

Mot			Caractère			Caractère n-gram			Tous		
Class	model	Score	Class	model	Score	Class	model	Score	Class	model	Score
1	MNB	0.991931	1	SVM	0.997030	1	SVM	0.996881	1	SVM	0.99725
2	SVM	0.990396	2	LR	0.994604	2	BNB	0.995941	2	LR	0.99710
3	LR	0.984208	3	BNB	0.994505	3	LR	0.995099	3	RF	0.99695
4	RF	0.974851	4	RF	0.993762	4	RF	0.994653	4	BNB	00.99640
5	DT	0.950446	5	KNN	0.991287	5	GB	0.990297	5	GB	0.99525
6	BNB	0.948911	6	GB	0.989356	6	MNB	0.986436	6	MNB	0.99115
7	GB	0.936089	7	MNB	0.986386	7	KNN	0.986337	7	DT	0.98465
8	KNN	0.260050	8	DT	0.984010	8	DT	0.985644	8	KNN	0.98165

- **Etape 01 :**

On trouve que les résultats de Suppression de ponctuation et de Suppression de Mot Vide sont encore très similaires et globalement les classificateurs diminuent un petit peu par rapport à l'étape précédente par exemple LR il était 99,710% il est devenu 99,6139%.

Tableau 17: Résultats de l'Etape 01 3émé corpus

Suppression de Ponctuation											
Mot			Caractère			Caractère n-gram			Tous		
Class	Model	Score	Class	Model	Score	Class	Model	Score	Class	Model	Score
1	MNB	0.991931	1	SVM	0.997030	1	SVM	0.996881	1	SVM	0.997079
2	SVM	0.990396	2	LR	0.994604	2	BNB	0.995941	2	LR	0.996139
3	LR	0.984208	3	BNB	0.994505	3	LR	0.995099	3	RF	0.994653
4	RF	0.975743	4	RF	0.993416	4	RF	0.995050	4	BNB	.992822
5	BNB	0.948911	5	MNB	0.986386	5	MNB	0.986436	5	MNB	0.987277
Suppression de Mots Vides											
Mot			Caractère			Caractère n-gram			Tous		

Class	Model	Score	Class	Model	Score	Class	Model	Score	Class	Model	Score
1	MNB	0.991931	1	SVM	0.997030	1	SVM	0.996881	1	SVM	0.997079
2	SVM	0.990396	2	LR	0.994604	2	BNB	0.995941	2	LR	0.996139
3	LR	00.984208	3	BNB	0.994505	3	LR	0.995099	3	RF	0.994505
4	RF	0.974802	4	RF	0.994257	4	RF	0.994802	4	BNB	0.992822
5	BNB	0.948911	5	MNB	0.986386	5	MNB	0.986436	5	MNB	0.987277

- **Etape 02 :**

Si on compare les résultats de ce corpus avec ceux du corpus précédent on trouve qu'ils sont très proches et ont qualifié les mêmes classificateurs (SVM, LR, BNB) et la même représentation (tous)

Tableau 18: Résultats de l'Etape 02 3émé corpus

Suppression de ponctuation et de Mot Vides											
Mot			Caractère			Caractère n-gram			Tous		
Class	model	Score	Class	model	Score	Class	model	Score	Class	model	Score
1	SVM	0.999752	1	SVM	0.999851	1	SVM	0.999851	1	SVM	0.999851
2	LR	0.997822	2	LR	0.997673	2	LR	0.998020	2	LR	0.999752
3	BNB	0.941287	3	BNB	0.992178	3	BNB	0.994950	3	BNB	0.991881
0 .979620			0,996567			0,997607			0,998921		

5.1.2. Discussion des résultats obtenus

- Les classificateurs discriminants obtiennent généralement de meilleurs résultats comme LR et SVM.
- DT donne toujours le même score même si le corpus devient plus grand.
- RF fonctionne mieux que AD car comme nous avons déjà dit dans le chapitre trois que forte aléatoire c'est une amélioration à arbre de décision.
- KNN donne généralement de mauvais résultats avec caractère et n-gramme de caractère Mais ses résultats sont acceptables en cas de mini-corpus et dans la représentation par mots.
- GB est mieux approprié pour les petits corpus par rapport aux autres classificateurs
- Les résultats du mini corpus complet : GB avec 99.5%, SVM avec 99.16%, LR avec 99.16%.
- Les résultats corpus moyen complet : SVM avec 99.98%, LR avec 99.98%, BNB avec 99 .28%.

- Les résultats Grand corpus complet : SVM avec 99.98%, LR avec 99.97%, BNB avec 99 .18%.
- Plus le volume de corpus est grand plus sera élevée la valeur du score mais nous avons remarqué que le moyen et le grand corpus donnent presque les mêmes résultats.

- La figure 42 représente la comparaison des résultats obtenus dans les quatre représentations, on a remarqué que la combinaison donne de meilleurs résultats sachant que les résultats sont très similaires.

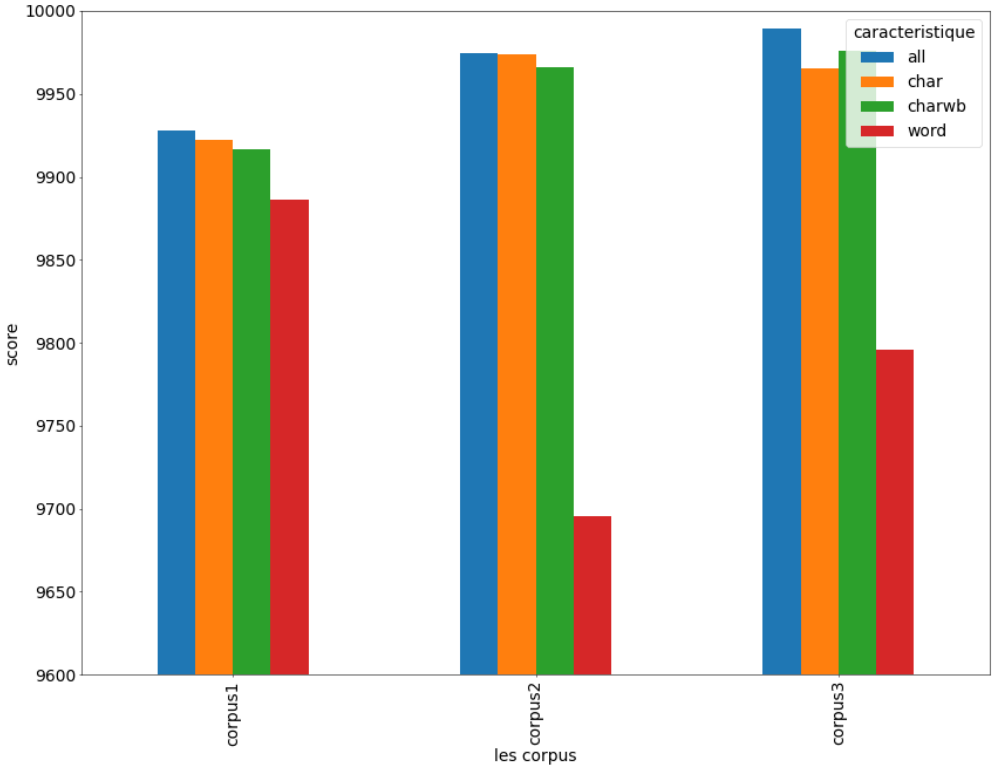


Figure 42: Résultats de filtrage de représentation

- Le tableau 19 représente les résultats du 2eme corpus des quatre prétraitements proposés, on remarque que les meilleurs résultats sont de suppression Ponctuation +mot vide.

Tableau 19 : Résultats de filtrage de prétraitement

	SVM	**LR**	**BNB**
aucun	0.997745	0.996176	0.993922
Ponctuation	0.997745	0.996176	0.993922
Mot vide	0.991667	0.991667	0.993922
Ponct +MV	0.999804	0.999804	0.992843

Dans le but de choisir le meilleur corpus nous avons dessiné le graphe suivant (figure 43) qu'illustre que les corpus moyen et grand sont très adjacents.

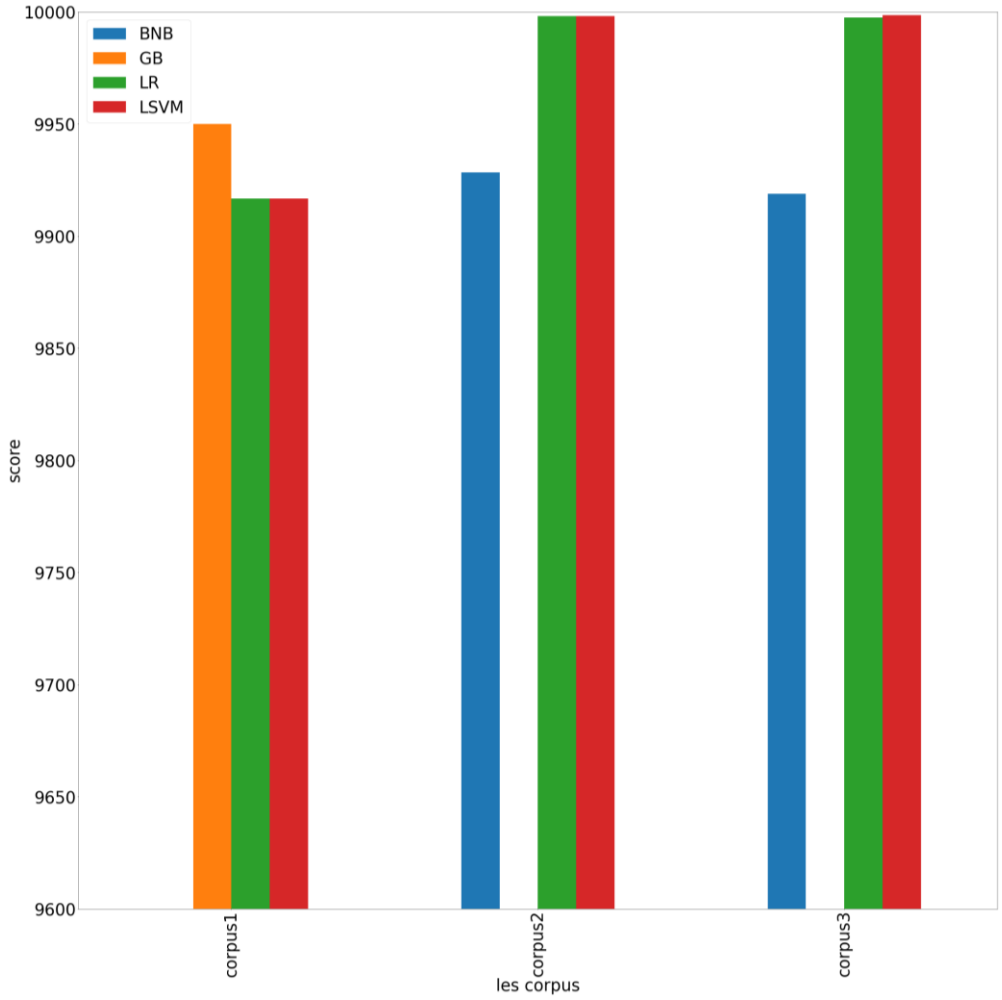


Figure 43: Résultats de filtrage de modèle

5.1.3. Output de processus

Ce résultat représente output de 1er filtrage statistique et input de 2eme filtrage :

- ✓ Les meilleurs classificateurs : SVM avec 99.98%, LR avec 99.98%, BNB avec 99 .28%.
- ✓ La meilleure représentation vectorielle : la combinion (Mot, caractère et n-gramme de caractère).
- ✓ Le meilleur prétraitement : suppression de mots vide et de ponctuation.
- ✓ Le meilleur corpus : moyen corpus (51000 phrases).

5.2. Les résultats d'Etude de robustesse du système

Nous avons continué avec le corpus moyen (51000) évalué par les trois classificateurs (SVM, LR, BNB) en appliquant la configuration combinaison. En plus, nous avons introduit deux nouvelles méthodes : Centroide de lettre et n-gramme CT. Nous avons expérimenté avec plusieurs tailles de phrase en commençant par une phrase contenant un mot de 3 caractères et en augmentant la taille peu à peu jusqu'à une phrase de taille quelconque.

5.2.1. Résultat détaillé

Tableau 20: Résultats de Test de robustesse (%)

méthode	Taille	1M /3C	1M /5C	2M/3C	2M/5C	2M	3M/3C	3M	4M	complet
n-gramme CT [1]	scores	81	74	81	80	74.16	83.25	79.08	82.83	93
	Temps	0 .46	1 .3	1.3	2 .3	2.072	1.964	2.970	3.8247	57.91
Centroide de lettre [2]	scores	50	57.25	59.25	62.6	64.41	65.91	68	70.25	91
	temps	0.039	0.0441	0.045	0.048	0.048	0.047	0.051	0.0542	0.1014
SVM [3]	scores	79.25	69	76	75	77.58	81.50	85.33	88.08	99.98
	temps	0.001	0.0017	0.002	0.002	0.0031	0.0034	0.004	0.0061	0.0709
LR [4]	scores	76.50	67	72	74	75.25	76.91	81.33	83.66	99.98
	temps	0.021	0.0271	0.021	0.022	0.033	0.026	0.029	0.0312	0.090
BNB [5]	scores	53.25	68	73	78	73.58	79.91	79.16	81.91	95.08
	temps	0.368	0.3449	0.37	0.34	0.375	0.373	0.371	0.3856	0.4775

5.2.2. discussion des résultats obtenus

[1] : score (74% → 93%) et le temps (0 .46s → 57.91s)

cette méthode donne de bons résultats(81%) même en cas de 1Mot de 3 lettre avec un temps d'exécution suffisant de 0 .46s On note qu'il maintient ses bonnes performances entre 74% et 82% jusqu'à 4mot mais le temps augmente progressivement pour atteindre 15s avec 16mot et 57s avec le teste complet et ceci est considéré comme long .

[2] : score (50% → 91%) et le temps (0.0397 s → 0.1014 s)

La performance est considérée comme la plus faible pour les mots courts avec 50 % de cas d'un seul mot et augmente au 68% avec 4 mot ensuite cela devient acceptable avec une phrase.

[3] : score (69% → 97%) et le temps (0.0010s → 0.07s)

Au terme de temps il est considéré comme le plus rapide et ça donne aussi de bons résultats avec les mots courts mais l'avantage est que le temps est court.

[4] : score (67% → 97%) et le temps (0.021 s → 0.09025s)

Les résultats de ce classificateur sont très proches au présentant.

[5] : score (53% → 95%) et le temps (0.34s → 0.47s)

Les résultats de ce classificateur sont médiocres par rapport à un mot de 3 caractères 53% mais sa performance commence à augmenter avec l'augmentation du nombre des mots jusqu'à ce qu'elle arrive à 95% dans une phrase.

5.2.3. Output final

- ✓ Nous avons choisi les classificateurs : SVM 79.25%, LR 76.50%, n-gramme CT 81% pour les utiliser dans l'application grâce à leur robustesse même avec les petites unités.

6. Comparaison de la méthode proposée avec les travaux connexes

Tableau 21: Comparaison de la méthode proposée avec les travaux connexes.

méthodes	Auteur	Corpus	Langues	Résultats	
				Les leurs	Les nôtres
SVM	Seungbeom Kim 2007[22]	1000 documents sur Wikipédia	Anglais	98,22%	99.98%
MNB	Jhamtani	Reuters	danois, allemands, espagnols,	99.22%	98.31%

[22] Seungbeom Kim, Jongsoo Park ,2007 Automatic Detection of Character Encoding and Language

	2014[23]		français, italiens, néerlandais, norvégiens, portugais et suédois ,anglais		
DT[24]	Häkkinen 2001	Environ 17 000 noms distincts	l'anglais, le finnois, l'espagnol et l'allemand.	90.9%	98.50%
Centroide de lettre	Hidayet Takçı 2012 [28]	né (291 K), ang (108 K),fra (108 K), alle (171 K), ital (99 K), por (107 K), espa (107 K), sué (91 K), Turc (109 K).	Néerlandais anglais, français, allemand italien, portugais, espagnol, suédois et turc	97.50%	91%
LR	Ben King 2014[25]	corpus DSL (DSLCC)	Bosniaque/Croate/Serbe, Indonésien/ Malais, Tchèque/Slovaque, Portugais Espagnol Anglais	87,04%	99.98%

7. Présentation de l'application

En premier temps l'utilisateur peut faire juste des tests aléatoires proposés par notre système.

[23] : Jhamtani, H., Bhogi, S. K., & Raychoudhury, V. (2014). Word-level Language Identification in Bi-lingual Code-switched Texts

[24] Häkkinen, J., & Tian, J. (2001). N-gram and Decision Tree Based Language Identification for Written Words.

[25] Ben King, Dragomir Radev, and Steven Abney. 2014. Experiments in sentence language identification with groups of similar languages.

Figure 44: Accueil d'Identificateur

Le résultat de test aléatoire apparaît dans une fenêtre.

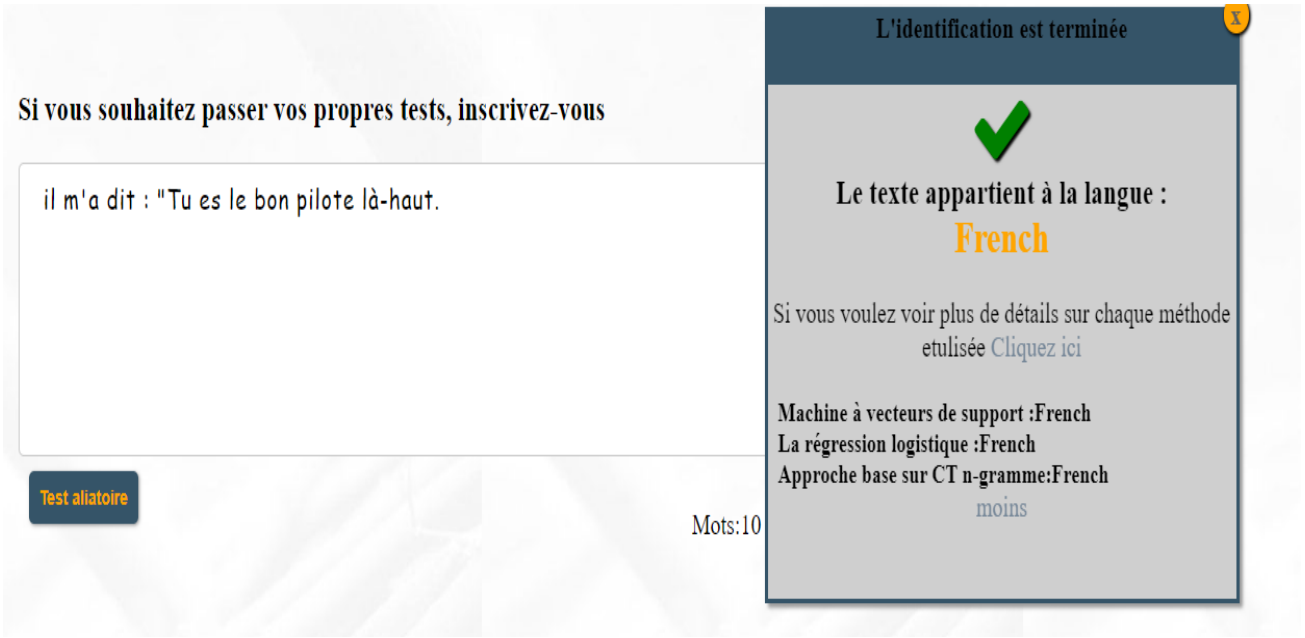


Figure 45: Resultats de Test Aléaotoire

Apres si vous Souhaitez passer vos propres tests inscrivez-vous sur notre Plateforme vous pouvez identifier des textes ou bien des documents de votre choix.

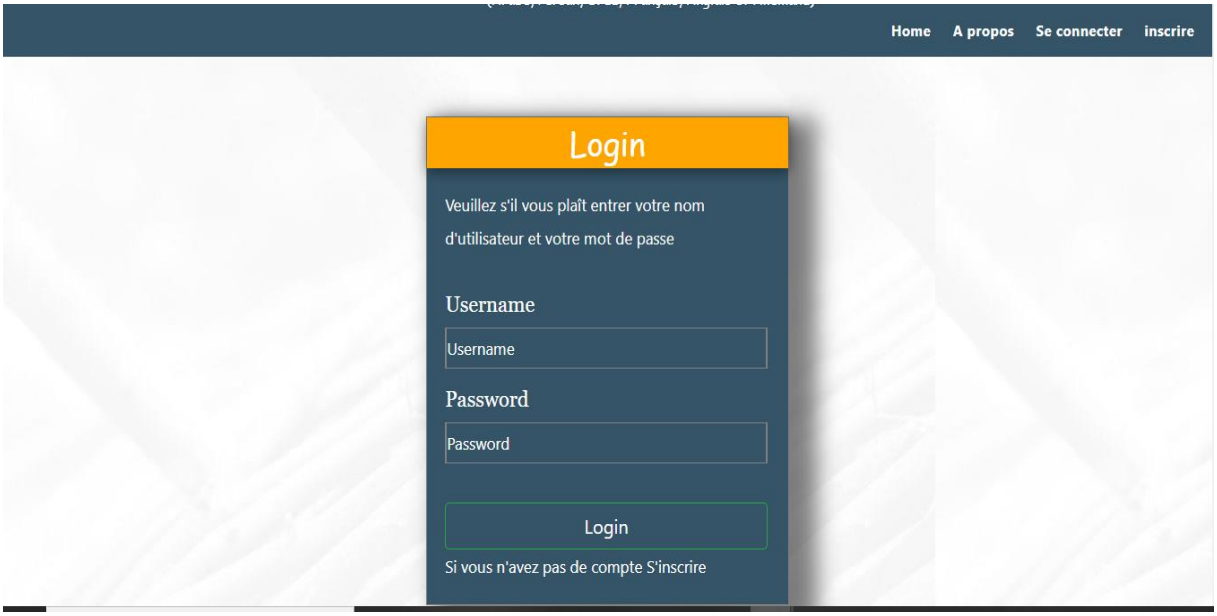


Figure 46: Espace de Connexion

Dans votre inscription l'email doit etre valide sinon on ne met pas la forme correcte dans le système , il va la signaler comme erreur.

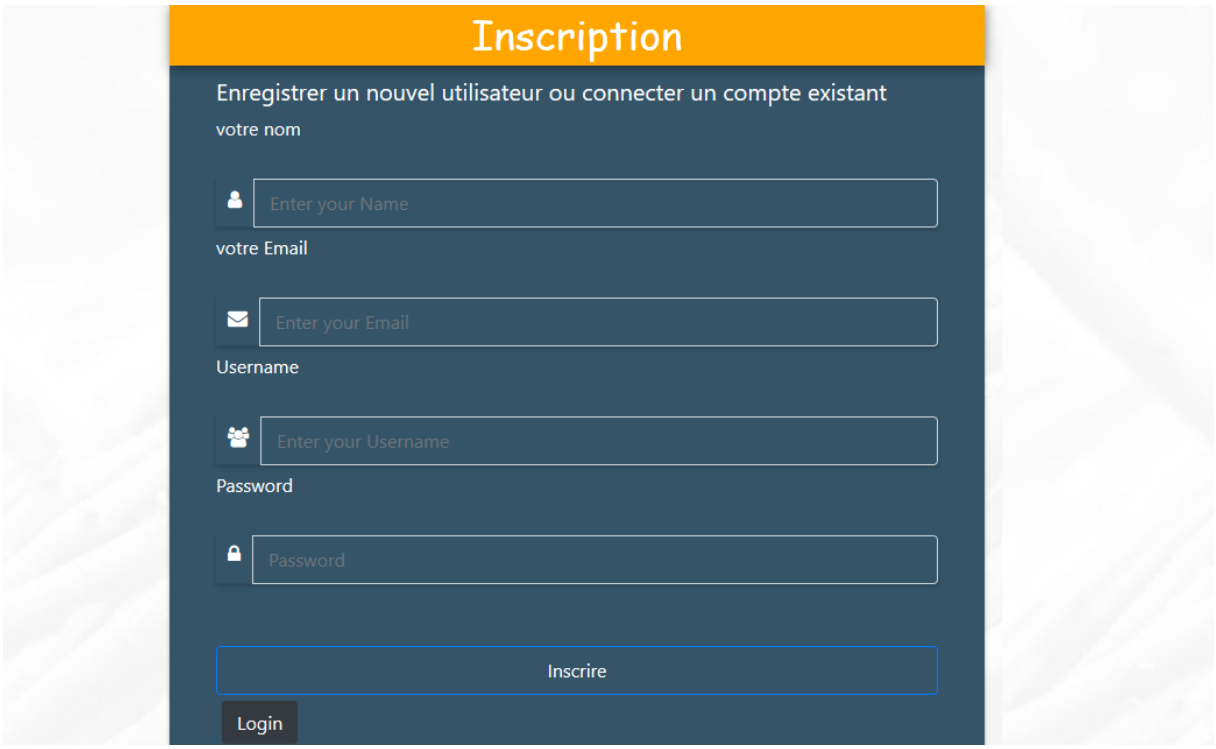


Figure 47: Espace d'Inscription

Apres l'inscription, vous povez passer votre test, nous avons par exemple entre la phrase suivante " لا تكن متفرجا في هذه الدنيا « et comme nous avons interessé à la langueur de phrase notre système compte le nombre de mots.

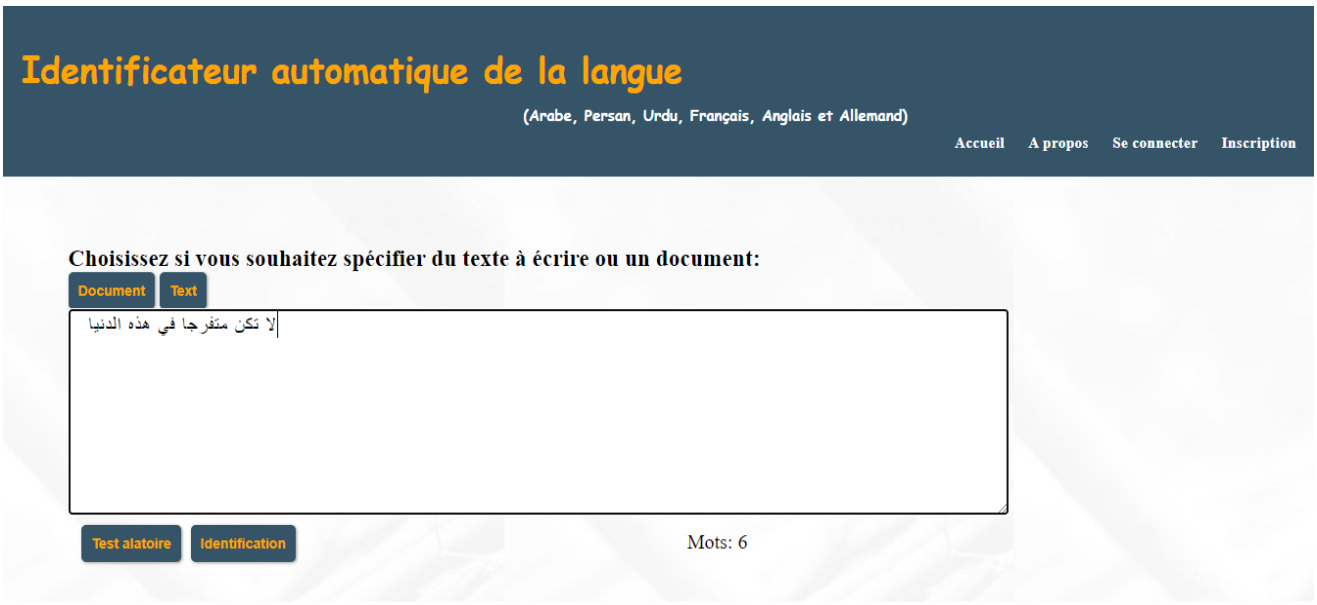


Figure 48: Exemple d'Identification de la Langue

Le système retourne le résultat **'arabe'** malgré que les trois méthodes donnent des résultats totalement différents car on prend en considération dans notre système la précision de chaque méthode comme un mesure de pondération tel que :

- Machine à vecteurs de support : 79
- La régression logistique : 76
- Approche CT n-gramme : 81

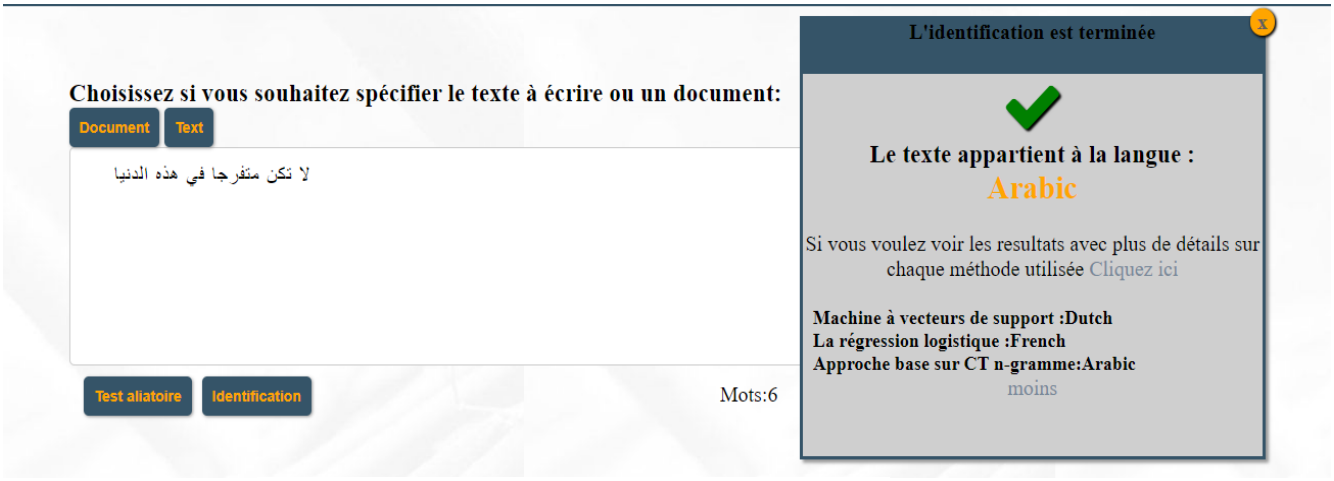


Figure 49 : Exemple plus détaillé

Conclusion et perspective

CONCLUSION GENERALE

L'objectif de notre projet de fin d'étude est de réaliser un système d'identification de la langue qui identifie six langues de différentes morphologies (Ourdou, Persan, Français, Anglais, Allemand, Arabe) et chacune de ces langues a des difficultés. Le travail est divisé en deux phases : une phase de recherche bibliographique et théorique qui a mis ensemble trois chapitres et une phase pratique qui a englobé deux chapitres.

Notre solution est considérée comme une solution hybride entre les approches statistiques et approches linguistiques. L'approche statistique se base sur un ensemble de méthodes de classification tels que : Machine à Vecteurs de Support (SVM) et Bernoulli naïve Bayes (BNB) et Multinomial Naïve Bayes (MNB) (qui sont basées sur algorithme de naïve Bayes) et Algorithme des k-voisins les plus proches(KNN) et Régression Logistique (LR) et Arbre de décision(DT) et Forêt aléatoire (RF) et gradients Boosting (GB). L'approche linguistique se base sur les lettres et n-gramme de lettres.

Nous avons fait une comparaison de la performance des techniques d'apprentissage utilisées pour détecter automatiquement la langue. Et ça nous a permis de connaître les limites de chacune des méthodes utilisées et le/la meilleur(e) (méthode, représentation vectorielle, prétraitement, taille de corpus) pour atteindre le meilleur résultat et améliorer l'efficacité du système d'identification. Les résultats obtenus sont les suivants :

- ✓ Les meilleurs classificateurs : SVM avec 99.98%, LR avec 99.98%, BNB avec 99 .28%
- ✓ La meilleure représentation vectorielle : la combinions (Mot, Caractère, N-gramme de caractère).
- ✓ Le meilleur prétraitement : suppression de mot vide et de ponctuation.
- ✓ Le meilleur corpus : moyen corpus (51000 phrases).

A ce niveau nous avons utilisé les outputs de l'étape précédente et les approches linguistiques pour évaluer l'étape suivante qui est basée sur taille de fragment et la vitesse de méthode.

La plus grande exactitude (Accuracy) 81% est obtenue avec la méthode de n-gramme de CT avec un fragment d'un seul mot de trois caractères.

Pour finaliser le travail nous avons utilisé les trois meilleures méthodes (n-gramme de CT 91%, SVM 99.98%, LR99.98%) obtenues pour développer notre application web.

Perspective

Du fait que l'identification automatique de la langue est un domaine très large, plusieurs perspectives s'offrent à ce projet :

- Documents multilingues : Les documents multilingues sont des documents qui contiennent du texte dans plus d'une langue.
- Pauvreté de la recherche dans Traitement automatique de la langue ourdou et persan : stemmer, lemmatisation et segmentation.
- Les recherches suivantes se concentrent beaucoup plus sur les langues proches et rares.
- Essayer d'hybrider les classificateurs de machine Learning avec les caractéristiques linguistiques par exemple utiliser le classificateur SVM avec les n-grammes les plus distinctifs de la langue.
- Être flexible avec les données textuelles provenant de différents systèmes de codage peuvent être traitées et converties au format UTF-8.
- Nous pouvons améliorer notre système, en ajoutant un identificateur de langue vocale.
- Créez un système de filtre des langages pour lesquels nous n'avons pas de données d'apprentissages, mais qui peuvent néanmoins être rencontrés par un système LID donc c'est très intéressant de faire un filtrage pour des langues spécifiques.

Références

[1] K. R. Beesley., language identifier: A computer program for automatic natural-language identification of on-line text, languages at Crossroads: Proceedings of the 29th Annual l Conferenceof the American Translators Association: pp. 47-54, , 12-16 October 1988,.

[2] W. B. &. T. J. M. Cavnar, N-Gram-Based Text Categorization, Las Vegas, In Proceedings of SDAIR-94, Third Annual Symposium on Document Analysis and Information, 1994.

[3] A. Moodley, Language Identification With Decision Trees: Identification Of Individual Words In The South African Languages Thesis, South Africa: A dissertation submitted in partial fulfilment for the degree of Bachelor of Science Honours in Computing, 2016.

[4] S. Mustonen, Multiple Discriminant Analysis in Linguistic Problems., 4, 37–44: Statistical Methods in Linguistics, 1965.

[5] E. M. Gold, Language Identification in the Limit., 10(5): Information and Control, (1967).

[6] G. R. L. a. G. R. Doddington, Automatic language identification, Air Force Rome Air Development Center: Technical report RADC-TR-74-200, 1974.

[7] S. Johnson., Solving the problem of language recognition., University of Leeds: Technical report, School of Computer Studies, 1993. .

[8] G. Grefenstette, Comparing two language identification schemes, Rome: 3rd International conference on Statistical Analysis of Textual Data, 1995, Dec 11–13.

[9] P. S. a. J. C.Reynar, Language identification: Examining the issues, .: In Proceedings of the 5th Symposium on Document Analysis and Information Retrieval, 1996.

[10] J. M. Prager, «Linguini: Language identification for multilingual documents.,» In HICSS ’99: Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences .IEEE Computer Society., vol. 2, n° %1Washington,

DC, USA, p. page 2035, 1999.

[11] I. M. Y. O. A. C. Y. Suzuki, «A language and character set determination method based on n-gram statistics.,» ACM Trans. Asian Lang. Inf.Process., Vols. %1 sur %21 (,, n° %13, p. 269–278., 2002.

[12] M. P. a. L. Padro, Comparing methods for language identication, Barcelona, Spain: Proceedings of the XX Congreso de la Sociedad Espanola para el Procesamiento del Lenguage Natural, 2004.

[13] H. Hammarstrom., «A naive theory of affixation and an algorithm for extraction.,» In Proceed ings of the Eighth Meeting of the ACL Special Inter est Group on Computational Phonology and Morphology at HLT-NAACL 2006, Vols. %1 sur %2 NewYork City, USA, Junes, n° %1 Association for Computational Linguistic, p. pages 79–88, 2006.

[14] C.-C. S. A. Ng, Improved letter weighting feature selection on arabic script language identification, pp. 150–154.: In: Proc. ACIIDS, 2009.

[15] T. L. N. Gottron, A comparison of language identification approaches onshort, query-style texts., p. 611–614: In: Proc. ECIR, 2010.

[16] F. M. V. Winkelmolen, Statistical language identification of short texts., pp. 498–503: In: Proc. ICAART, 2012.

[17] P. S. V. S. a. H. t. I. Canasai Kruengkrai, Language identification based on string kernels., Beijing, China.: In Proceedings of the 5th Internamation Technologies (ISCIT2005), pages 896899,, 2005.

[18] M. L. a. T. Baldwin., Cross-domain feature selection for language identification., Citeseer: In In Proceedings of 5th International Joint Conference on Natural Language Processing. , 2011.

[19] M. S. X. Y. a. H. L-F. Zhai, Gish Discrim inatively trained language models using support vector machines for language identification, San Juan, Puerto Rico, pp. 1-6: IEEE Odyssey 2006: The Speaker and Language Recog nition Workshop, June, 2006.

[20] A. E. Z. S. M. Amine, Automatic language identification: an alternative unsupervised approach using a new hybrid algorithm., 7 (1), 94–107: Int. J Comput. Sci. Appl. , 2010.

[21] R.-M. Bali., Automatic Identification of Close Languages–Case Study: Malay and In donesian, 2(2):126–133: ECTI Transaction on Computer and In formation Technology, 2006.

[22] S. M. a. M. Dras, Automatic Language Identification for Persian and Dari texts, Bali,Indonesia, May: In Proceedings of the 14th Conference of the Pacific Association for Computational Linguistics (PACLING 2015), pages 59–64,, 2015.

[23] N. M. a. D. B. Nikola Ljubesic, Language indentification: How to distinguish similar languages?, pages 541–546. IEEE: In Information Technology In terfaces, 2007. ITI 2007. 29th International Conference on, 2007. .

[24] C.-R. H. a. L.-H. Lee, Contrastive Approach towards Text Source Classification based on Top-Bag-Word Similarity, .: ., 2008.

[25] J.-L. Rouas, Caracterisation et identification, Universit´e Toulouse III, 2005.

[26] Y. K. Hakan Ceylan, Language Identification of Search Engine Queries, ?: ?, 2009.

[27] M. L. a. T. Baldwin, Accurate Language Identification of Twitter Messages, ?: ?, 2014.

[28] T. G. Hidayet Takçı, A high performance centroid-based classification approach, Turkey: Department of Computer Engineering, GYTE, Kocaeli 41400, 2012.

[29] M. Majlis, Yet Another Language Identifier, France: Association for Computational Linguistics, 2012.

[30] M. L. M. Z. T. B. TS Jauhiainen, «Automatic Language Identification in Texts: A Survey,» Journal of Artificial Intelligence Research, vol. 65, n° %1., p. 675–782, 2019.

[31] G.-I. Kikui, «Identifying the Coding System and Language of Online Documents on the Internet,» chez COLING ’96, Copenhagen, Denmark, In Proceedings of the 16th

International Conference on Computational, 1996, p. pp. 652–657.

[32] T. S. M. T. O. &. W.-H. C. Mandl, Language Identification in Multi-lingual Web-Documents, pp. 153–163, Klagenfurt, Austria: In Proceedings of the 11th International Conference on Applications of Natural Language to Information Systems (NLDB 2006), 2006.

[33] V. S. P. P. C. &. K. A. Simaki, Identifying the Authors' National Variety of English in Social Media Texts, In Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP 2017), pp. 671–678: Varna, Bulgaria, 2017.

[34] P. Henrich, Language Identification for the Automatic Grapheme-to-phoneme Conversion of Foreign Words in a German Text-to-speech System., pp. 2220–2223, Paris, France: In First European Conference on Speech Communication and Technology, 1989.

[35] S. Langer, Natural Languages and the World Wide Web, .: Bulletin de Linguistique, 2001.

[36] L. G. Windisch, Language identification using global statistics of natural languages, .: Proceedings of the 2nd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence (SACI), 2005.

[37] M. D. Rau, Language Identification by Statistical Analysis., Monterey: Master's thesis, NavalPostgraduate School, 1974.

[38] S. R. A. K. A. N. S. K. B. S. &. R. P. Banerjee, A Hybrid Approach for Transliterated Word-Level Language Identification: CRF with Post Processing Heuristics., pp. 54–59, Bangalore, India: In Proceedings of the Sixth Workshop of the Forum for Information Retrieval Evaluation (FIRE 2014), 2014.

[39] T. V. J. J. &. V. S. Vatanen, Language Identification of Short Text Segments with N-gram Models, pp. 3423–3430, Valletta, Malta: In Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010), 2010.

[40] E. Sterneberg, Language Identification of Person Names Using Cascaded SVMs.,

Uppsala University, Uppsala.: Bachelor's Thesis, 2012.

[41] J.-C. F. V. &. W.-R. C. Marcadet, A Transformation-based Learning Approach to Language Identification for Mixed-lingual Text-to-speech Synthesis., pp. 2248–2251, Lisbon, Portugal.: In Proceedings of the 6th Interspeech and 9th European Conference on Speech Communication and Technology (EUROSPEECH), 2005.

[42] Y. Y. J. C. M. Z. Y. &. W. J. Chen, Identifying Language Origin of Person Names with N-grams of Different Units., Vol. 1, pp. 729–732, Toulouse, France.: In Proceedings of the 2006 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006),, 2006.

[43] J.-L. C. Y.-N. C. M. S. F. K. &. W. J.-L. You, Identifying Language Origin of Named Entity with Multiple Information Sources., 1077–1086.: Audio, Speech, and Language Processing, IEEE Transactions on, 16(6),, 2008.

[44] W. &. D. S. Adouane, Identification of Languages in Algerian Arabic Multilingual Documents, pp. 1–8, Valencia, Spain: In Proceedings of The Third Arabic Natural Language Processing Workshop (WANLP 2017), 2017.

[45] R. &. K. M. Rehůřek, Language Identification on the Web: Extending the Dictionary Method., pp. 357–368, Mexico: In Proceedings of the 10th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2009), 2009.

[46] W. E. H. A.-S. L. H. N. &. D. M. Salloum, Sentence Level Dialect Identification for Machine Translation System Selection., pp. 772–778, Baltimore, USA.: In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), , (2014).

[47] A. K. Singh, Study of Some Distance Measures for Language and Encoding Identification., Sydney,Australia.: In Proceedings of the Workshop on Linguistic Distances, pp. 63–72, (2006).

[48] M. Š. I. Z. K. &. P. S. Martinc, Author Profiling Gender and Language Variety Prediction—Notebook for PAN at CLEF 2017., Dublin, Ireland.: In Working Notes Papers of CLEF 2017 Evaluation Labs and Workshop, , (2017).

[49] F. Sebastiani, Machine learning in automated text categorization, Italy: Consiglio Nazionale delle Ricerche, March 2002 .

[50] M. L. a. T. Baldwin, Cross-domain feature selection for language identification, VIC 3010, Australia: Department of Computer Science and Software Engineering University of Melbourne, VIC 3010, Australia, 2011.

[51] O. Choayb, Classification automatique de textes, mesila algerie: unviverite UNIVERSITE DE M'SILA mémoire master , 2014.

[52] K. K. e. al, Text Classification Algorithms: A Survey, Charlottesville , USA,: Department of Systems and Information Engineering, University of Virginia, 230avril 2019.

[53] H. Saif, M. Fernández, Y. He et H. Alani, On stopwords, filtering and data sparsity for sentiment analysis of twitter, Reykjavik, Iceland: In Proceedings of the Ninth International Conference on Language Resources and Evaluation, 26–31 May 20 (LREC 2014).

[54] D.D.Lewis, An evaluation of phrasal and clustered representations on a text categorization task, university of chicago: Center for information and language studies, 1992.

[55] H. Shimodaira, Text Classification using Naive Bayes Hiroshi Shimodaira, Édimbourg: University of Edinburgh., January-March 2020.

[56] S. M. a. S. Scott, Feature engineering for text classification, .: presented at International Conference On Machine Learning, , 1999.

[57] B. A. Asma Al-Omari, «ARABIC LIGHT STEMMER (ARS),» Journal of Engineering Science and Technology , vol. 9, n° %16, pp. 702 - 717 , 2014.

[58] G. J. A. C.Buckley, Automatic query expansion using SMART, ornell University, Ithaca: Department of Computer Science, 1994 .

[59] F. A. MAHAMMED, APPROCHES D'APPRENTISSAGE AUTOMATIQUE POUR LA DÉTECTION DU SPAM WEB : EXPLORATION DE DIVERSES CARACTÉRISTIQUES, Montréal : UNIVERSITÉ DU QUÉBEC À MONTRÉAL ,

2018 .

[60] L. B. ,. G. M. Lamia Belguith Hadrich, STAr : un Système de Segmentation de Textes Arabes basé sur l'analyse contextuelle des signes de ponctuations et de certaines particules, Tunisie : Laboratoire LARIS-Faculté des Sciences Economiques et de Gestion BP 1088, 3018-Sfax–, 2005.

[61] C. A. &. A. B. H. Lamia Hadrich Belguith,) MASPAR : De la segmentation à l'analyse syntaxique de textes arabes, Faculté des Sciences Economiques et de Gestion de SfaxB.P. 1088, 3018 - Sfax –: TUNISIE, 2007.

[62] M. CHERAGUI, une Analyse Morphologique de la langue arabe basée sur l'Aide Muticritere à la Décision, Algérie: Université d'Adrar, 2012.

[63] S. K. M. A. M Ali, Pattern Based Comprehensive Urdu Stemmer and Short Text Classification., Pakistan: Department of Computer Engineering, Bahria University Islamabad, 2017.

[64] J. N. M. I. Gupta V, Design and development of rule based inflectional and derivational Urdu stemmer 'Usal', (ABLAZE), pp. 7–12: In: Proceedings of IEEE international conference on futuristic trends on computational analysis and knowledge management , (2015) .

[65] W. K. &. D. C. Ali Daud, Urdu language processing : asurvey, USA: Springer Science+Business Media Dordrecht , 2016.

[66] S. I. Abdul Jabbar, A survey on Urdu and Urdu like language stemmers and stemming techniques, Multan Pakistan: Department of Computer Science, Institute of Southern Punjab, 2016.

[67] S. J. Alireza Mokhtaripour, Introduction to a new Farsi stemmer, .: CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management, 2006 .

[68] H. S. J. M. I. Mehrnoush Shamsfard, A Set of Fundamental Tools for Persian Text Processing, Valletta: Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23, May 2010.

[69] M. M. F. O. M Tashakori, Bon: First Persian Stemmer, .: ., 2002.

[70] F. M. J. D. AH Jadidinejad, Evaluation of Perstem: A Simple and Efficient, .: Proceedings of the 10th cross-language evaluation forum conference on Multilingual information access evaluation: text retrieval experiments, 2009.

[71] R. J. Somayye Estahbanati, A New Stemmer For Farsi Language, Iran: CSI International Symposium on Computer Science and Software Engineering (CSSE), 2011.

[72] D. Francoeur, Machines à vecteurs de supportUne introduction, .: Cahiers Mathématiques de l'Université de Sherbrooke, 2010.

[73] S. &. P. J. Kim, Automatic Detection of Character Encoding and Language, Stanford University, 2007.

[74] M. TAFFAR, Initiation à l'Apprentissage Automatique, algerie : Université de Jijel, Support de Cours pour étudiants en Master en Intelligence Artificielle.

[75] J. QUINLAN, Induction of Decision Trees, Boston: Kluwer Academic Publishers, ,81-106, 1986.

[76] S. J. A Mokhtaripour, Introduction to a new Farsi stemmer, .: ., 2006 .

[77] R. Rakotomalala, Arbres de Décision, france: Université Lumière Lyon 2 , 2005.

[78] K. M. a. G. B. Kumar, «Language Identification from small text samples,» The Journal of Quantitative Linguistics,, vol. 13, n° %11, pp. 57-80 , 2006.

[79] N. Horning, Random Forests : An algorithm for image classification and generation of continuous fields data sets, USA: International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences , 2010.

[80] A. N. a. A. Knoll, Gradient boosting machines, a tutorial,, Garching, Munich, Germany : 2Department of Informatics, Technical University Munich, dec 2013 VOL7.

[81] J. D. A. K. E. G. U. H. L. A. &. L. M. Capstick, «A system for supporting cross-lingual information retrieval,» A system for supporting cross-lingual information

retrieval. Information Processing & Management , vol. 2, n° %136, pp. 275-289, March 2000.

[82] Y. M. C. A. M. a. S. T. N. Chew Y. Choong, «Optimizing n-gram Order of an n-gram Based Language Identification Algorithm for 68 Written Languages,» journal.icter.org, vol. 02, n° %102, pp. 21 - 28, 2009.

[83] H. T. a. ø. So÷ukpınar, Letter Based Text Scoring Method for Language Identification, allemand: Verlag Berlin Heidelberg, 2004.

[84] E. B. a. R. C. Y.K. Muthusamy, Automatic language identification: a review/tutorial, IEEE Signal Processing Magazine: Vol. 11, 1994, pp. 33 - 41.

[85] R. JALAM, Apprentissage automatique et catégorisation de textes multilingues, LYON: UNIVERSITÉ LUMIÈRE , 2003.

[86] B. M. a. M. J. Silva, Language identification in web pages, In Proceedings of the 2005 ACM Symposium on Applied Computing (SAC '05): page 764, 2005.

[87] M. Ali, S. Khalid, M. Haneef, W. Iqbal, A. Ali et G. Naqvi, «A Rule based Stemming Method for Multilingual Urdu,» Journal international des applications informatiques, vol. 134 , n° %18, p. 0975 – 8887, 2016.

[88] W. A. ,. U. B. ,. X. W. SA Khan, A Light Weight Stemmer for Urdu Language: A Scarce Resourced Language, Mumbai: Proceedings of the 3rd Workshop on South and Southeast Asian Natural Language Processing (SANLP), 2012, p. pages 69–78.

[89] P. McNamee, Language identification: a solved problem suitable for undergraduate instruction, .: Journal of Computing Sciences in Colleges, February 2005.

[90] R. U. I. M. W. A. Tayyaba Fatima, Morphological and Orthographic Challenges in Urdu Language Processing: A Review, Lahore, Pakistan: Department of Computer Science, COMSATS Institute of Information Technology, Juin 2018.

[91] Assas-Band, an Affix-Exception-List Based Urdu Stemmer, Suntec, Singapore: Proceedings of the 7th Workshop on Asian Language Resources, pages 40–47, 6-7 August 2009.

[92] N. J. I. M. Vaishali Gupta, Rule Based Stemmer in Urdu, Rajasthan, India: Apaji Institute, Banasthali University, 2014.