



Royaume du Maroc

Université Abdelmalek Essaâdi

Faculté des Sciences et Techniques

D'Al-Hoceima

المملكة المغربية

جامعة عبد المالك السعدي

كلية العلوم والتكنولوجيات

الحسمة



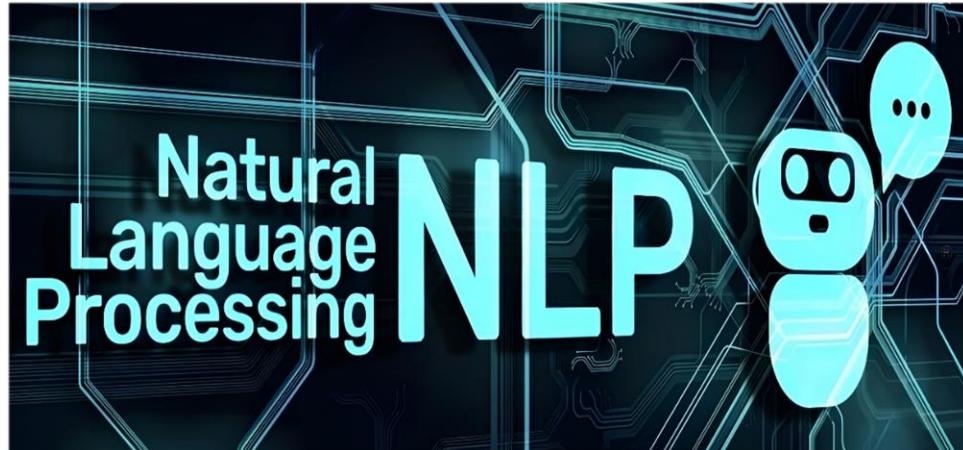
Département : Mathématique et Informatique

Licence Sciences et Techniques

Spécialité : Mathématique-Informatique

## Classification supervisée de texte

basée sur Machine Learning



*Rapport projet de fin d'étude*

Projet réalisé par :

AMEZIANE Mohamed

Encadré par :

Pr. SOUFI Adil

Soutenu le : 20 / 06 / 2023

Devant les membres de jury :

- Pr. FAHIM Mohamed (Président)
- Pr. SRAI Abdelaziz (Examinateur)
- Pr. SOUFI Adil (Encadrant)

Année Universitaire : 2022/2023

“Imagine two worlds, one with you and one without you. What’s the difference between the two worlds? Maximize that difference. That’s the meaning of your life.”

Kai-Fu Lee

## Dédicaces

---

*Je dédie ce décente travail*

### *A mes chers parents*

*Les mots me manquent pour exprimer toute la reconnaissance, la fierté et le profond amour que je vous porte pour les sacrifices que vous avez consentis pour ma réussite. Que vous trouvez ici le témoignage de mes attachements, ma reconnaissance, ma gratitude et mon respect. Que Dieu vous préserve une bonne santé et longue vie.*

### *A mes chers frère et sœur*

*J'espère avoir atteint le seuil de vos espérances. Que ce travail soit l'expression de ma profonde affection. Je vous remercie pour le soutien moral et l'encouragement que vous m'avez accordé.*

*Je vous souhaite un brillant avenir.*

### *A mon encadrent*

*Ma gratitude s'adresse également à M. Soufi Adil pour la confiance qu'il m'a témoignée en acceptant d'encadrer mon travail et pour les précieux conseils qu'ils n'ont cessés de me prodiguer tout au long de ce projet.*

### *A mes amis*

*En honneur à la grande amitié qui nous unit, je vous remercie du fond de mon cœur pour les moments agréables si précieux passés ensemble. Je vous aime.*

### *A Toute personne qui vient de chercher son nom ici*

*Un mot final pour vous tous, j'espère que j'ai été à la hauteur de vos espérances et sachez que jamais je pourrais oublier ce que vous avez fait pour moi, petit qu'il soit ou grand.*

*AMEZIANE Mohamed*

## **Remerciements**

---

Je rends grâce à Dieu, le Tout-puissant de m'avoir donné le courage, la volonté, la patience, la santé et la bénédiction tout au long de mon parcours universitaire jusqu'à l'aboutissement de ce projet de fin d'études et que grâce à Lui ce travail a pu être réalisé. Je tiens à remercier vivement toutes les personnes, qui, de près ou de loin, se sont impliquées dans la réalisation de ce projet, tant par leur présence, leurs conseils, leur disponibilité et leur soutien opérationnel, que professionnel.

Je souhaite exprimer ma plus grande gratitude à mon encadrant pédagogique, M. SOUFI Adil, de l'FST d'Al-Hoceima, pour ses remarques expertes qui ont permis d'améliorer la qualité de mon travail. J'espère sincèrement avoir été à la hauteur de la confiance qu'il m'a témoignée et que ce travail répondra à ses attentes. Mes remerciements vont également à l'ensemble des membres du jury pour leur intérêt pour mon travail et l'honneur qu'ils font en acceptant de le juger.

Enfin, un remerciement spécial à mes parents et à mes frère et sœur pour le soutien et les sacrifices. Ainsi qu'à toute personne ayant contribué, de près ou de loin, à la réussite de ce travail.

## Résumé

---

Avec l'accroissement de la quantité de données textuelles générées par toute l'humanité dans le monde, notamment sur l'internet, il devient de plus en plus impératif de les traiter de manière intelligente afin d'en extraire diverses formes de connaissances. Mon objectif dans ma recherche est de développer des modèles d'apprentissage capables d'induire automatiquement des représentations du langage humain, en particulier la détection de langue et l'analyse des sentiments dans des phrases précises, afin de résoudre de multiples tâches linguistiques de haut niveau.

Beaucoup de recherches ont été menées dans la mise en œuvre de technologies de traitement du langage naturel. Ma recherche vise à développer des algorithmes capables d'analyser les sentiments et de détecter les langues en se basant sur l'intelligence artificielle.

Ce sujet de recherche au sein de *Faculté des Sciences et Techniques d'Al-Hoceima* m'a fait ressortir l'esprit de recherche qui m'a amené à trouver des solutions adéquates, quel que soit le problème auquel j'ai confronté.

**Mot clés** : Traitement de langage naturel, Intelligence Artificielle, NLP, Deep Learning, Machine Learning.

## **Abstract**

---

With the increasing amount of textual data generated by all humankind worldwide, especially on the Internet, it is becoming more and more imperative to process it intelligently in order to extract various forms of knowledge. The aim of my research is to develop learning models capable of automatically inducing representations of human language, in particular language detection and sentiment analysis in specific sentences, in order to solve multiple high-level linguistic tasks.

Considerable research has gone into the implementation of natural language processing technologies. My research aims to develop algorithms for sentiment analysis and language detection based on artificial intelligence.

This research topic at the Faculty of Science and Technology in Al-Hoceima has brought out the spirit of research in me, which has led me to find appropriate solutions, whatever the problem I've faced.

**Key words :** Natural Language Processing, Artificial Intelligence, NLP, Deep Learning, Machine Learning.

# Sommaire

---

<b>DEDICACES.....</b>	I
<b>REMERCIEMENTS.....</b>	II
<b>RESUME.....</b>	III
<b>ABSTRACT.....</b>	IV
<b>SOMMAIRE.....</b>	V
<b>TABLE DE FIGURES .....</b>	VIII
<b>TABLE DES EQUATIONS .....</b>	ERROR! BOOKMARK NOT DEFINED.
<b>LISTE DES ABREVIATIONS .....</b>	IX
<b>INTRODUCTION GENERALE .....</b>	1
<b>CHAPITRE I : CONTEXTE DE PROJET.....</b>	3
I.    INTRODUCTION .....	4
II.   PRÉSENTATION DU PROJET.....	4
1.    Problématique.....	4
2.    Objectifs .....	4
III.  PLANIFICATION DES TÂCHES.....	5
1.    Tableau des taches .....	5
2.    Diagramme de GANTT .....	6
IV.   CONCLUSION.....	7
<b>CHAPITRE II : ETAT DE L'ART .....</b>	9
I.    INTRODUCTION .....	9
II.   GÉNÉRALITÉS SUR L'INTELLIGENCE ARTIFICIELLE .....	9
1.    Qu'est-ce que l'Intelligence Artificielle ? .....	9
2.    Apprentissage Automatique (Machine Learning) .....	10
i.    Qu'est-ce que l'apprentissage automatique (ML)? .....	10
ii.   Quand utiliser l'apprentissage automatique ?.....	10
3.    Apprentissage Profond (Deep Learning) .....	11
i.    Qu'est-ce que l'apprentissage profond ?.....	11
ii.   Quand utiliser l'apprentissage profond ?.....	11
III.  TYPES D'APPRENTISSAGE.....	12
1.    Apprentissage supervisé .....	12
2.    Apprentissage non supervisé .....	12
3. <i>Apprentissage par renforcement</i> .....	13

IV.	ALGORITHMES DE MACHINE LEARNING SUPERVISÉ.....	13
1.	La régression linéaire simple .....	13
2.	La régression linéaire simple .....	14
3.	La régression polynomiale.....	14
4.	La régression logistique.....	15
5.	L'arbre de décision .....	16
6.	Les forêts aléatoires .....	16
7.	Machine à vecteurs de support .....	17
8.	KNN .....	17
9.	Naïve Bayes.....	18
V.	CLASSIFICATION DE TEXTE.....	19
1.	Traitemet de Langue Naturelle .....	19
2.	Algorithme choisi pour ce problème .....	20
3.	Fonctionnement d'algorithme.....	20
i.	Préparation des données.....	20
ii.	Extraction de features .....	21
iii.	Apprentissage .....	21
iv.	Classification .....	21
v.	Evaluation.....	21
4.	Prétraitement de texte .....	22
i.	Normalisation .....	22
ii.	Tokenisation.....	22
iii.	Suppression des stopwords .....	23
iv.	Stemming / Lemmatisation.....	23
VI.	CONCLUSION.....	24
<b>CHAPITRE III : CONCEPTION DE PROJET</b>	.....	<b>26</b>
I.	INTRODUCTION .....	26
II.	LANGUAGE DE MODÉLISATION UML .....	26
III.	DIAGRAMME DE CAS D'UTILISATIONS .....	26
IV.	DIAGRAMME DE SÉQUENCE.....	27
1.	Diagramme de séquence d' <b>Analyseur de Sentiments</b> .....	28
2.	Diagramme de séquence de <b>Détecteur de langue</b> .....	29
V.	ARCHITECTURE GLOBALE D'APPLICATION .....	30
VI.	CONCLUSION.....	31
<b>CHAPITRE IV : IMPLEMENTATION ET REALISATION</b>	.....	<b>33</b>
I.	INTRODUCTION .....	33
II.	OUTILS DE DÉVELOPPEMENT .....	33

1.	Langage Python .....	33
2.	Installation de Python .....	34
3.	Visual Studio Code .....	36
4.	Installation de VS Code .....	36
III.	BIBLIOTHÈQUES UTILISÉES : .....	38
1.	Pandas.....	38
2.	NumPy.....	38
3.	Scikit-Learn (sklearn).....	38
4.	NLTK (Natural Language Tool Kit) .....	39
5.	Custom Tkinter & Tkinter .....	39
6.	Joblib .....	39
7.	BS4 (Beautiful Soup 4).....	39
8.	Re .....	39
9.	Unidecode.....	40
IV.	IMPLÉMENTATION.....	40
1.	Création d'un projet python.....	40
2.	Acquisition et préparation de données.....	41
i.	Chargement de Dataset .....	41
ii.	Normalisation .....	41
iii.	Tokenisation.....	42
iv.	Suppression des stopwords .....	42
v.	Stemmisation .....	43
vi.	Application sur la Dataset.....	43
3.	Phase d'apprentissage de la machine.....	43
i.	Division de Dataset en celle de test et celle d'apprentissage .....	43
ii.	Vectorization (Bag-of-words) .....	43
iii.	Apprentissage du modèle.....	44
iv.	Score et Performance .....	44
V.	INTERFACES PRINCIPALES .....	46
VI.	DÉMONSTRATION .....	48
VII.	CONCLUSION.....	49
<b>CONCLUSION GENERALE .....</b>	<b>50</b>	
<b>REFERENCES.....</b>	<b>51</b>	

# Table de Figures

---

Figure 1 : Tableau des tâches -----	5
Figure 2 : Diagramme de GANTT-----	7
Figure 3 : Les piliers de recherche de l'IA -----	9
Figure 4 : Données Structurées vs. Non Structurées -----	11
Figure 5 : IA, ML et DL -----	12
Figure 6 : Types d'apprentissage -----	12
Figure 7 : Régression linéaire simple -----	14
Figure 8 : Régression polynomiale -----	15
Figure 9 : Fonction sigmoïde ou courbe en S -----	16
Figure 10 : Arbre de décision-----	16
Figure 11 : SVM-----	17
Figure 12 : K-Nearest Neighbors -----	18
Figure 13 : Diagramme de cas d'utilisations -----	27
Figure 14 : Diagramme de séquence pour " Sentiment Analyzer"-----	28
Figure 15 : Diagramme de séquence pour "Language Detecter"-----	29
Figure 16 : L'architecture générale de "Sentiment Analyzer" -----	30
Figure 17 : L'architecture générale de "Language Detecter" -----	31
Figure 18 : Logo de Python-----	33
Figure 19 : Etape I d'installation de Python -----	34
Figure 20 : Etape II d'installation de Python -----	34
Figure 21 : Etape III d'installation de Python -----	35
Figure 22 : Etape IV d'installation de Python -----	35
Figure 23 : Logo de VS Code -----	36
Figure 24 : Etape I d'installation de VS Code-----	36
Figure 25 : Etapes d'installation de VS Code -----	37
Figure 26 : Interfaces Principaux (Sombre/Clair) -----	46
Figure 27 : Interface de "Sentiment Analyzer" -----	47
Figure 28 : Interface de "Language Detecter" -----	47
Figure 29 : Démonstration de "Sentiment Analyzer" -----	48
Figure 30 : Démonstration de "Language Detecter" -----	49

## Liste des abréviations

---

- ◆ **NLP** : Natural Language Processing
- ◆ **TLN** : Traitement de Langue Naturelle
- ◆ **TALN** : Traitement Automatique de Langue Naturelle
- ◆ **ML** : Machine Learning
- ◆ **DL** : Deep Learning
- ◆ **AI** : Artificial Intelligence
- ◆ **SVM** : Support Vector Machine
- ◆ **KNN** : K Nearest Neighbor
- ◆ **HTML** : HyperText Markup Language
- ◆ **XML** : Extensible Markup Language
- ◆ **CSV** : Comma Separated Values
- ◆ **ASCII** : American Standard Code for Information Interchange

# Introduction Générale

---

Le Traitement du Langage Naturel (NLP : Natural Language Processing) a toujours suscité mon intérêt en raison de la fascination qu'exercent le cerveau humain et nos capacités cognitives. Il est véritablement remarquable de constater à quel point nous sommes capables de communiquer des informations, des pensées complexes et des émotions avec si peu d'efforts, ce qui rend la tentative de reproduire cette capacité dans des machines encore plus intéressante. Bien que nous progressions rapidement dans le domaine de l'intelligence artificielle (IA), nous n'avons pas encore atteint ce stade. Il se peut que réussir le test de Turing, qui évalue la capacité d'une machine, à présenter un comportement intelligent équivalent à celui d'un être humain, ne soit pas suffisant. Peut-on réellement reproduire tous les aspects d'un être humain avec une machine ?

À mesure que la quantité de données textuelles non structurées produites par l'être humain augmente chaque jour, il devient de plus en plus nécessaire de les traiter intelligemment et d'en extraire différents types de connaissances.

A ce propos, mon projet fin d'étude vise à développer deux modèles d'apprentissage automatique, le premier est consacré pour l'analyse des sentiments et le deuxième pour la détection des langues. Ces deux programmes représentent une application de la classification supervisée des textes, et montrent également l'ensemble du processus par lequel la machine passe, en commençant par le nettoyage de la dataset, puis choix d'un algorithme approprié, ensuite l'entraînement d'algorithme sur cet ensemble de données, pour finalement obtenir un modèle efficace capable d'avoir de haut scores. Mon objectif principal est donc de montrer les différentes étapes de traitement de texte lors de l'obtention de résultats.

Les travaux de ce travail seront successivement détaillés dans ce manuscrit qui est constitué de 4 chapitres :

- **Le 1<sup>er</sup> chapitre :** est consacré à la présentation de contexte de projet y compris la problématique et la solution envisagée.

- **Le 2<sup>ème</sup> chapitre :** représente une bibliographie concise sur quelques généralités de ce projet y inclut l'intelligence artificielle, la classification supervisée de texte, pré-traitement de texte...etc.
- **Le 3<sup>ème</sup> chapitre :** présente une étude de la conception du projet, en introduisant les langages de programmations utilisés.
- **Le 4<sup>ème</sup> chapitre :** est consacré à la description des différentes étapes suivies pour réaliser le programme final, puis la description du résultat obtenu.

Et finalement, nous récapitulons par une conclusion générale.

# *Chapitre I*

---

---

## *Contexte de Projet*

---



---

### Contenu de chapitre :

- Présentation du projet
  - Définition de la problématique
  - Planification de projet
- 



# Chapitre I : Contexte de Projet

## I. Introduction

Le présent chapitre est un chapitre introductif visant à définir le contexte de projet et ses objectifs spécifiques, ainsi que l'approche méthodologique de travail et la planification des tâches.

## II. Présentation du projet

### 1. Problématique

Le traitement du langage naturel (NLP) est une branche très importante de l'intelligence artificielle, qui se focalise sur la compréhension et le traitement du langage humain automatiquement. Son objectif est de permettre aux ordinateurs de comprendre, interpréter et produire du langage humain de manière fluide et pertinente.

En utilisant des techniques avancées telle que l'apprentissage automatique, le NLP ouvre la voie à de nombreuses applications pratiques, allant des assistants virtuels intelligents qui permettent de comprendre les requêtes et les commandes des utilisateurs, leur fournir des réponses pertinentes et les guider dans leurs interactions avec les systèmes informatiques, en passant par la traduction automatique et l'analyse de sentiments. Grâce aux progrès dans ce domaine, les machines peuvent désormais traiter et interagir avec le langage humain de manière de plus en plus naturelle et efficace.

La problématique centrale de ce projet vise à développer deux modèles de machine Learning capable d'analyser les sentiments et de détecter la langue. Ils s'agissent de deux applications de la classification supervisée des textes.

Dans cette perspective, le projet vise à analyser le sentiment d'un texte donné ou à détecter la langue d'un texte donné à l'aide d'une classification supervisée. L'algorithme utilisé dans ce projet est "*Naive bayes*".

### 2. Objectifs

Les objectifs principaux de ce projet sont :

- Collecte et préparation d'une dataset pour chaque modèle.
- Choix d'algorithme approprié à ce problème

- Entraînement du modèle d'analyse de sentiment en utilisant la dataset approprié. Même chose pour le modèle de détection de langue.
- Création d'une interface graphique qui facilite les tâches pour les utilisateurs.

### III. Planification des Tâches

La planification d'un projet est une étape clé de la phase d'avant-projet. Elle implique non seulement la définition d'un calendrier pour le projet, mais également la prévision du développement des activités tout au long de la période allouée au projet. En général, la planification d'un projet commence par sa division en plusieurs phases, l'estimation de sa durée, la détermination de la séquence des phases et l'allocation de ressources (financières et humaines), avec la participation de toutes les parties concernées.

#### 1. Tableau des tâches

Le tableau des tâches est un tableau qui permet d'établir un *planning de projet* bien organisé. Le tableau ci-dessous représente la planification du temps correspondant à ce projet :

Name	Begin date	End date
Exécution Complète du Projet	4/10/23	6/14/23
Apprendre les fondamentaux de Python	4/10/23	4/18/23
Phase de Recherche et d'Exploration	4/19/23	5/4/23
Collecte et Préparation des données	5/5/23	5/10/23
Développement du modèle	5/5/23	5/15/23
Implémentation et Inégration du modèle dans une interface développée	5/16/23	5/25/23
Redaction du Rapport	5/26/23	6/14/23

Figure 1 : Tableau des tâches

- Explication de chaque tâche
- **Apprendre les fondamentaux de Python :** en commençant par se familiariser avec les concepts fondamentaux de la programmation Python, puis en explorant des concepts tels que les fonctions, les modules, la gestion des fichiers ... etc. Enfin, comprendre la programmation orientée objet.

- **Phase de recherche et d'Exploration :** Tout d'abord, j'ai essayé de rassembler et d'étudier des documents universitaires, des articles... etc. appropriés. Ensuite, j'ai étudié les différents algorithmes et techniques d'apprentissage automatique couramment utilisés. Enfin, j'ai exploré les bibliothèques et les projets existants pour l'analyse des sentiments et la détection du langage.
- **Collecte et préparation des données :** Après avoir identifié un ensemble de données adapté à notre objectif, on commence à nettoyer les données et à assurer qu'elles sont labellisées de manière appropriée pour l'apprentissage, en effectuant finalement les tâches de prétraitement des données telles que *la normalisation, tokenisation et Stemming*.
- **Développement du modèle :** Après avoir étudié tous les algorithmes d'apprentissage automatique, on choisit l'algorithme approprié, on l'implémente dans le code et on l'entraîne sur l'ensemble de données préparé. Ensuite, vous devez affiner les modèles pour améliorer leur précision et leur performance au cas où ils obtiendraient de mauvais résultats lors de la validation croisée et de l'évaluation du modèle.
- **Implémentation et intégration des modèles dans une interface développée :** concevoir et développer une interface graphique et y intégrer les modèles construit puis tester le programme de manière approfondie et s'assurer qu'il gère différents types d'entrées et de scénarios.
- **Rédaction de rapport.**

## 2. Diagramme de GANTT

Le diagramme de Gantt est un outil fréquemment utilisé pour détailler les prévisions et le suivi du déroulement d'un projet. Il permet de visualiser clairement les différentes tâches ou étapes du projet, ainsi que leur planification et leur progression réelle au fil du temps. Le diagramme de Gantt offre une représentation graphique qui facilite la compréhension de la chronologie des activités et permet de suivre l'avancement du projet de manière visuelle.

En examinant le diagramme de Gantt suivant, on peut observer comment les différentes tâches du projet sont organisées dans le temps. Chaque tâche est représentée par une barre horizontale, dont la longueur correspond à sa durée prévue. Il est possible de repérer rapidement les retards ou les avancements par rapport au plan initial. Le schéma de Gantt illustré dans la Figure suivante présente le diagramme choisi pour représenter la planification du projet.

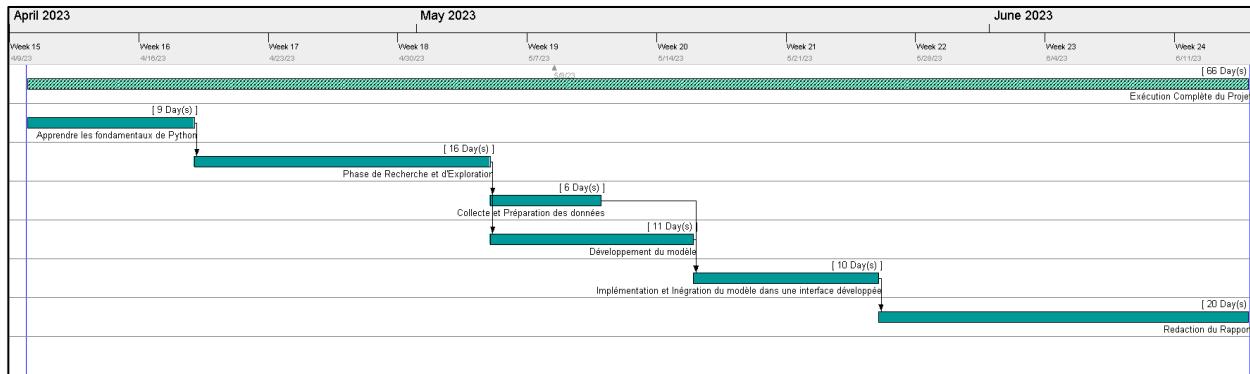


Figure 2 : Diagramme de GANTT

## IV. Conclusion

Le premier chapitre de ce rapport se concentre sur le cadre dans lequel le projet a été réalisé, une description détaillée du projet et de ses exigences, et la planification utilisée pour réaliser ce travail.

D'ailleurs, lorsqu'il s'agit de développer une solution avec une meilleure performance, l'aspect théorique devient un élément essentiel, raison pour laquelle nous avons consacré le chapitre suivant à une étude bibliographique de généralités sur l'Intelligence Artificielle.

# *Chapitre II*

---

---

## *Etat de l'Art*

---



### Contenu de chapitre :

- Intelligence Artificielle, Apprentissage Automatique et Apprentissage Profond
  - Apprentissage supervisée et non supervisée
  - Les Algorithmes d'apprentissage Automatique supervisé
  - Classification de texte
- 



# Chapitre II : Etat de l'Art

## I. Introduction

Le traitement du langage naturel représente une branche de l'intelligence artificielle dédiée à habiliter les machines à comprendre le langage humain. Son objectif principal est de développer des systèmes capables de donner un sens à un texte et d'exécuter des tâches de manière automatique.

Dans ce chapitre, j'aborderai d'abord l'historique de l'*intelligence artificielle*, puis je fournirai ce qu'est l'*apprentissage automatique* (*Machine Learning*) et l'*apprentissage profond* (*Deep Learning*). Nous explorerons ensuite les différents types d'apprentissage à savoir *Supervisé*, *Non supervisé* et *par renforcement*. Ensuite, je discuterai des algorithmes d'apprentissage supervisé. Enfin, je terminerai par l'étude de la classification des textes, l'algorithme employé dans ma solution et les étapes de prétraitement des textes.

## II. Généralités sur l'Intelligence Artificielle

### 1. Qu'est-ce que l'Intelligence Artificielle ?

Si beaucoup de jeunes pensaient que l'IA (*Intelligence artificielle*) était née dans les années 2000, ils se trompaient ! En réalité, c'est au cours de l'été 1956 que l'IA a été officiellement reconnue au *Darmouth College* (*New Hampshire, USA*) entre le 18 juin et le 17 août. Selon *John McCarthy*, *Marvin Minsky*, *Nathaniel Rochester* et *Claude Shannon*, l'IA part du principe que toutes les capacités cognitives humaines peuvent être refaites avec précision. Il sera possible de construire un système capable d'apprendre, de calculer, de mémoriser et pourquoi pas de faire des découvertes scientifiques. 1956 est la date à laquelle l'IA est considérée comme une science. Mais les travaux sur ce sujet ont commencé bien avant. La figure ci-dessous présente les piliers de l'IA.



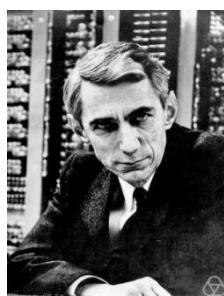
John McCarthy



Marvin Minsky



Nathaniel Rochester



Claude Shannon

Figure 3 : Les piliers de recherche de l'IA



Alan Turing

Selon le mathématicien *Alan Turing* en 1948 dans son article *Intelligent Machinery*, décrit mathématiquement les réseaux de neurones connectés aléatoirement et leur capacité à s'auto-organiser. Par ailleurs, le célèbre jeu appelé *Test de Turing* se résume ainsi : "si une personne, après quelques parties de questions-réponses avec un ordinateur, ne sait pas si elle a engagé une conversation avec un être humain ou une machine, on considère que l'ordinateur a réussi ce test"

Comme nous l'avons vu, l'IA n'est pas une science nouvelle. De plus, ce ne sont pas les robots ou les machines qui sont appelés IA, mais plutôt l'intelligence qu'ils utilisent. Il est indéniable que l'IA est une série de formules mathématiques basées principalement sur des statistiques et des probabilités. Elle fait référence au développement et à l'application de systèmes informatiques qui possèdent la capacité d'effectuer des tâches qui requièrent normalement l'intelligence humaine, tel que Minsky l'a définie.

L'IA se décline en deux parties, la première étant l'apprentissage automatique (Machine Learning) et la seconde l'apprentissage profond (Deep Learning).

## 2. Apprentissage Automatique (Machine Learning)

### i. Qu'est-ce que l'apprentissage automatique (ML) ?

L'apprentissage automatique est un sous-ensemble de l'intelligence artificielle qui utilise des algorithmes d'apprentissage statistique pour construire des systèmes qui ont la capacité d'apprendre automatiquement et de s'améliorer à partir d'expériences sans être explicitement programmés.

### ii. Quand utiliser l'apprentissage automatique ?

Lorsque vous cherchez à enseigner à un modèle comment effectuer une tâche, comme prédire un résultat ou découvrir un modèle à l'aide de données structurées. La figure 4 présente la différence entre les données structurées et non structurées par des exemples de notre vie quotidienne.

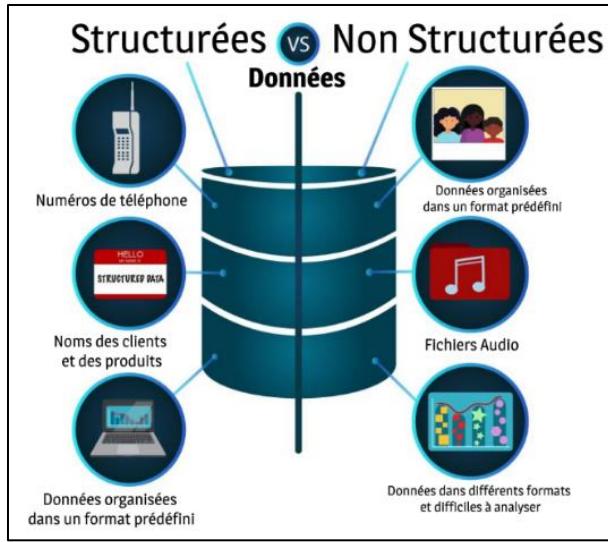


Figure 4 : Données Structurées vs. Non Structurées

Par exemple, Spotify vous construit une liste de lecture personnalisée basée sur vos chansons préférées et les données des autres utilisateurs qui partagent vos goûts.

### 3. Apprentissage Profond (Deep Learning)

#### i. *Qu'est-ce que l'apprentissage profond ?*

L'apprentissage profond est l'évolution de l'apprentissage automatique et des réseaux de neurones, qui utilise la programmation informatique avancée et l'apprentissage pour comprendre les modèles complexes cachés dans de grands ensembles de données. L'apprentissage profond consiste à comprendre comment le cerveau humain fonctionne dans différentes situations et à essayer de recréer son comportement.

#### ii. *Quand utiliser l'apprentissage profond ?*

L'apprentissage profond est utilisé pour accomplir des tâches complexes et former des modèles à l'aide de données non structurées. Par exemple, l'apprentissage profond est couramment utilisé dans la classification d'images telles que la reconnaissance faciale. Malgré les modèles d'apprentissage automatique puissent également identifier des visages, les modèles d'apprentissage profond sont plus précis.

Enfin, lorsque nous parlons d'IA, il s'agit soit de ML, soit de DL. Comme présenté dans la figure 5.

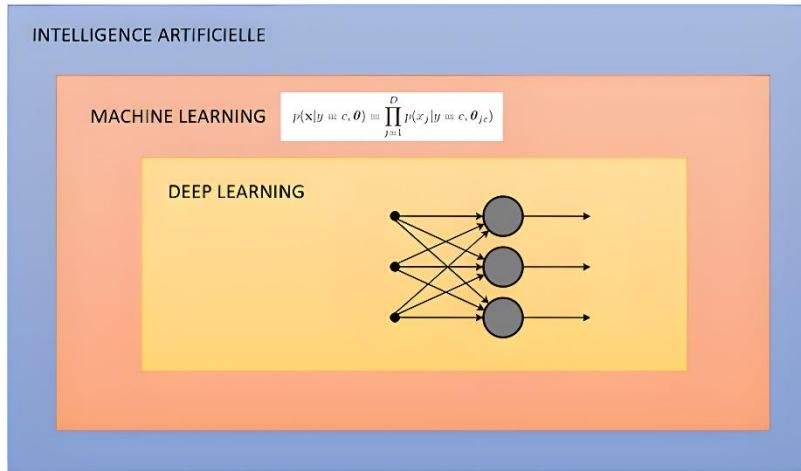


Figure 5 : IA, ML et DL

### III. Types d'apprentissage

Une machine peut éventuellement apprendre de trois manières différentes, à savoir :

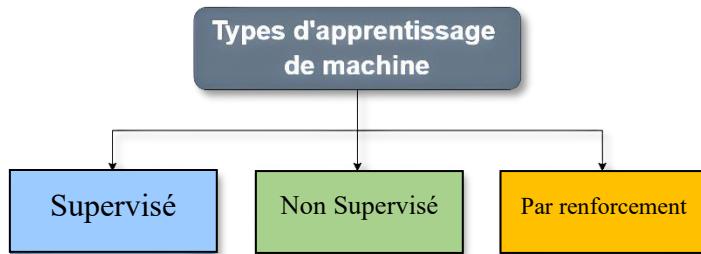


Figure 6 : Types d'apprentissage

#### 1. Apprentissage supervisé

L'apprentissage supervisé se définit par l'utilisation d'ensembles de données labellisés pour former des algorithmes capables de classer des données ou de prédire des résultats avec précision. Au fur et à mesure que les données d'entrée sont introduites dans le modèle, celui-ci ajuste ses poids jusqu'à ce qu'il soit correctement ajusté, ce qui se produit dans le cadre du processus de validation croisée.

Par exemple, l'ensemble de données utilisé pour former le modèle d'analyse des sentiments contient une colonne de texte, puis une colonne contenant la labellisation, ce qui signifie que chaque texte est labellisé comme positif ou négatif.

#### 2. Apprentissage non supervisé

L'apprentissage non supervisé se définit par l'utilisation des algorithmes d'apprentissage automatique pour examiner et catégoriser des ensembles de données non labellisés. Ces

algorithmes découvrent des modèles cachés dans les données sans avoir besoin d'une intervention humaine.

### 3. Apprentissage par renforcement

L'apprentissage par renforcement joue un rôle relativement mineur dans la formation à l'IA et s'apparente au dressage d'un animal. Lorsque l'animal adopte un comportement désiré, il reçoit une récompense. Il est défini comme un programme informatique interagit avec un environnement dynamique dans lequel il doit atteindre un certain objectif, sans qu'un enseignant lui dise explicitement s'il a atteint son but.

## IV. Algorithmes de Machine Learning Supervisé

Voici les algorithmes les plus courants connus dans le domaine de la ML supervisée :

### 1. La régression linéaire simple

La régression linéaire simple est une technique statistique utilisée cherche à établir la relation entre une variable indépendante(explicative) et l'autre dépendante(expliquée). Elle suppose qu'il existe une relation linéaire entre les deux variables, ce qui permet de prédire la valeur de la variable dépendante en fonction de la valeur de la variable indépendante. Voila l'équation que ce modèle est représenté par :

$$y = \alpha + \beta x + \epsilon \quad (1)$$

où

- $y$  est la variable dépendante.
- $x$  est la variable indépendante.
- $\alpha$  et  $\beta$  sont les coefficients de régression qui déterminent l'intercept et la pente de la droite de régression.
- $\epsilon$  est l'erreur résiduelle.

La méthode des moindres carrés est souvent utilisée pour estimer les coefficients de régression.

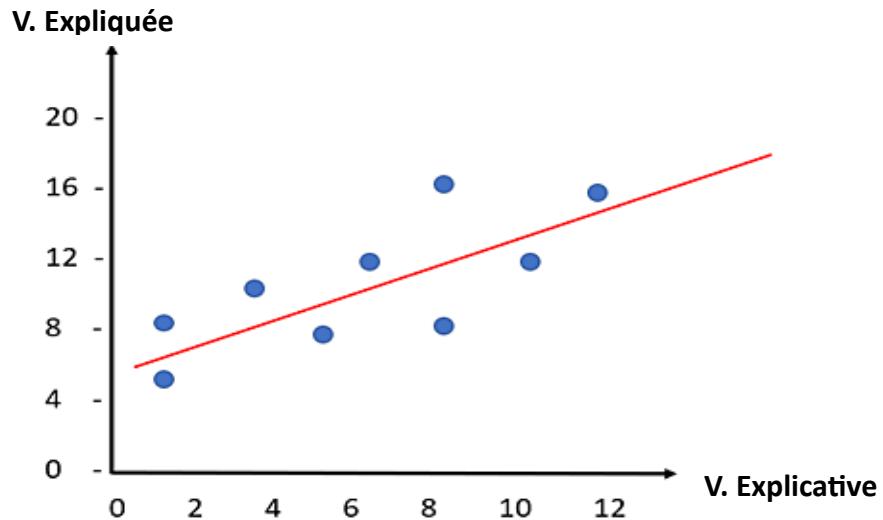


Figure 7 : Régression linéaire simple

## 2. La régression linéaire simple

La régression linéaire multiple est une extension de la régression linéaire simple qui permet de modéliser la relation entre une variable dépendante et plusieurs variables indépendantes. Elle est utilisée lorsque la relation entre la variable dépendante et les variables indépendantes ne peut pas être représentée par un modèle linéaire simple. Voilà l'équation que ce modèle est représenté par :

$$y = \alpha_0 + \alpha_1 x_1 + \cdots + \alpha_n x_n + \varepsilon = \alpha_0 + \sum_{i=1}^n \alpha_i x_i + \varepsilon \quad (2)$$

où

- $y$  est la variable dépendante.
- $x_1, x_2, \dots, x_n$  sont les variables indépendantes.
- $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_n$  sont les coefficients de régression correspondants.
- $\varepsilon$  est l'erreur résiduelle.

## 3. La régression polynomiale

La régression polynomiale est une extension de la régression linéaire qui permet de modéliser des relations non linéaires entre une variable dépendante et une variable indépendante. Elle consiste à ajuster un modèle polynomial à partir des données observées.

Voilà l'équation que ce modèle est représenté par :

$$y = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_n x^n + \varepsilon = \sum_{i=0}^n \alpha_i x^i + \varepsilon \quad (3)$$

où

- $y$  est la variable dépendante.
- $x$  est la variable indépendante.
- $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_n$  sont les coefficients de régression correspondants aux différents termes du polynôme.
- $x^2$  représente le carré de  $x$ ,  $x^3$  représente le cube de  $x$ , et ainsi de suite.
- $\varepsilon$  est l'erreur résiduelle.

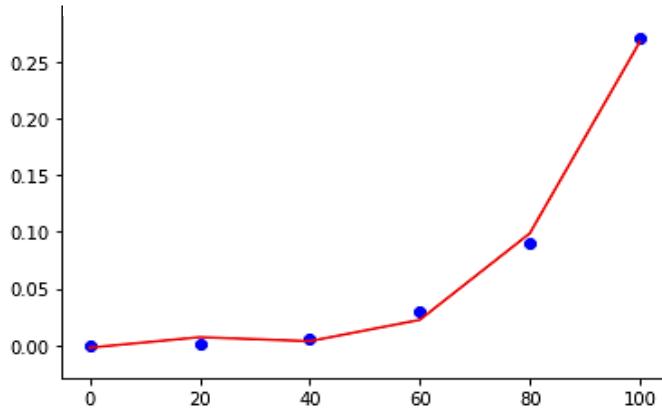


Figure 8 : Régression polynomiale

#### 4. La régression logistique

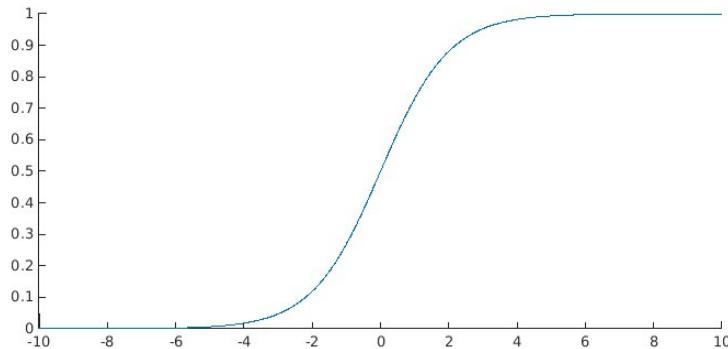
La régression logistique est un modèle statistique utilisé pour spécifier la relation entre une variable dépendante binaire et un ensemble de variables indépendantes.

Dans la régression logistique, le modèle est basé sur la fonction logistique (ou sigmoïde) qui transforme une combinaison linéaire des variables indépendantes en une probabilité. Voilà l'équation que ce modèle est représenté par :

$$p = \frac{1}{1 + e^{-z}} \quad (4)$$

où

- $p$  est la probabilité de la classe positive.
- $z$  est une combinaison linéaire des variables indépendantes et de leurs coefficients correspondants.



## 5. L'arbre de décision

L'arbre de décision nous permet à représenter un ensemble de choix sous forme d'un arbre, c'est une série de tests réalisés pour prédire un résultat. Il est toujours commencé par un nœud, et chaque extrémité des branches d'arbre présente la décision.

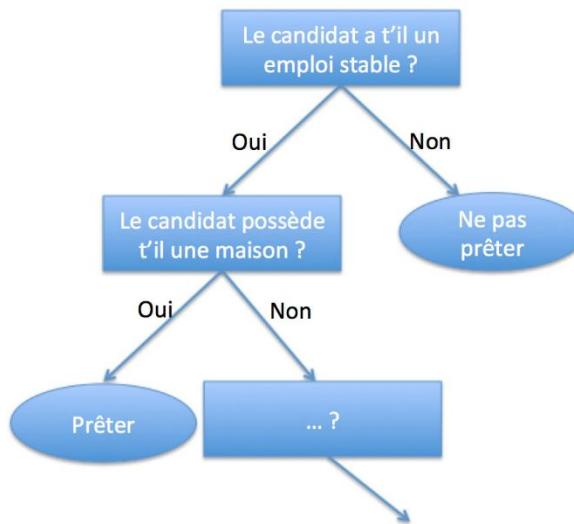


Figure 10 : Arbre de décision

## 6. Les forêts aléatoires

Une forêt est constituée d'arbres, c'est la même chose dans ce cas. La forêt aléatoire contient plusieurs arbres de décision. Son fonctionnement consiste à apprendre sur plusieurs arbres construits aléatoirement et entraînés sur des sous-ensembles contenant des données différentes. Enfin, on obtient plusieurs prédictions et la prédition finale obtenu par le calcul de la moyenne.

## 7. Machine à vecteurs de support

L'objectif principal de SVM est de trouver un hyperplan dans un espace de haute dimension qui sépare au mieux les données d'apprentissage appartenant à différentes classes. L'hyperplan est défini comme la frontière de décision qui maximise la marge entre les deux classes. Les vecteurs de support sont les exemples d'apprentissage les plus proches de l'hyperplan et jouent un rôle important dans la construction de la frontière de décision.

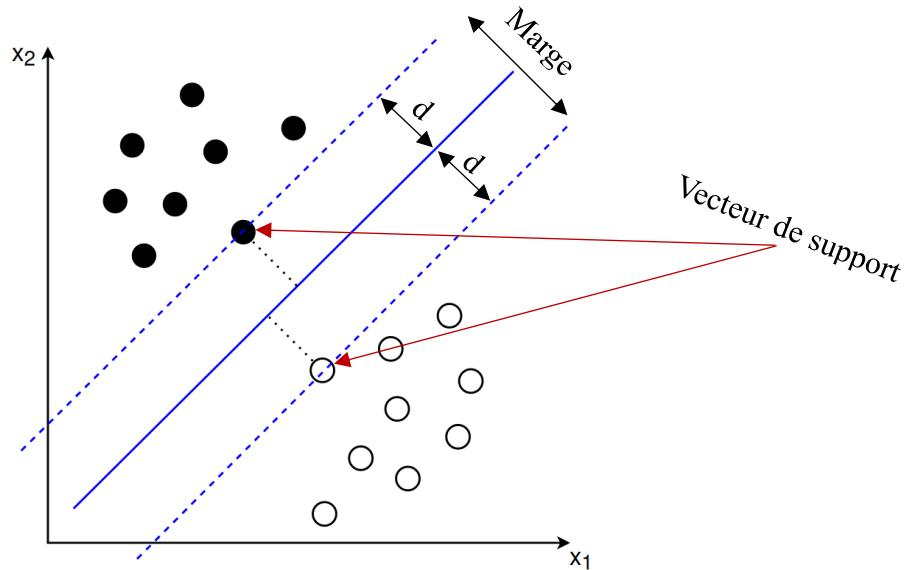


Figure 11 : SVM

## 8. KNN

KNN (*K-Nearest Neighbors* | *les K-Plus Proches*) est basé sur le principe que des exemples similaires ont tendance à appartenir à la même classe ou avoir des valeurs similaires. Chaque exemple dans la Dataset est représenté par des caractéristiques et un label (Cas de classification) ou une valeur cible (Cas de la régression), lors la phase d'apprentissage, KNN mémorise simplement la Dataset.

Pour faire une prédiction, KNN recherche les  $K$  exemples plus proches (plus similaires) du nouvel exemple dans l'espace des caractéristiques. Les classes des  $K$  voisins les plus proches sont ensuite utilisées pour attribuer un label de la classe ou effectuer une moyenne pour prédire une valeur.

Le choix de K est un paramètre important dans KNN. Une valeur trop petite de K peut conduire à une sensibilité excessive au bruit et aux valeurs aberrantes, par contre à une valeur trop grande de K peut entraîner une perte de détails et une confusion entre les classes.

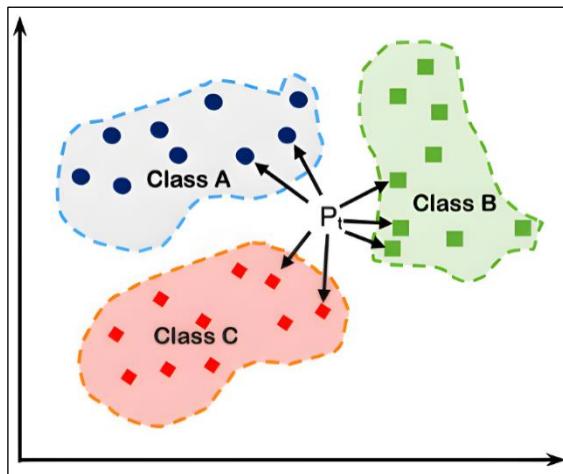


Figure 12 : K-Nearest Neighbors

## 9. Naïve Bayes

Naïve Bayes s'agit d'une technique de classification basée sur le théorème de Bayes avec une hypothèse d'indépendance entre les prédicteurs. En termes simples, *un classificateur « Naïve Bayes » suppose que la présence d'une caractéristique particulière dans une classe n'est pas liée à la présence d'une autre caractéristique*. En statistique, les classificateurs Naïve Bayes sont considérés comme des classificateurs probabilistes simples qui appliquent le théorème de Bayes.

Voici l'équation du théorème de Bayes :

$$P(c | x) = \frac{P(x | c) * P(c)}{P(x)} \quad (5)$$

où

- $P(c | x)$  représente la probabilité de la classe compte tenu des caractéristiques observées.
- $P(x | c)$  est la probabilité d'observer les caractéristiques compte tenu de la classe.
- $P(c)$  est la probabilité préalable de la classe c.
- $P(x)$  est la probabilité préalable des éléments d'entrée x.

Le tableau ci-dessous présente chaque algorithme et son catégorisation

*Table 1 : Les algorithmes de régression et de classification*

Algorithme	Régression	Classification
Régression linéaire simple	X	
Régression linéaire multiple	X	
Régression polynomiale	X	
Régression logistique		X
Arbre de décisions	X	X
Forêts aléatoire	X	X
SVM		X
KNN		X
Naïve Bayes		X

## V. Classification de texte

### 1. Traitement de Langue Naturelle

Le traitement du langage naturel, également connu sous le nom de traitement automatique du langage naturel (ou NLP en anglais | Natural Language Processing), est un domaine de l'informatique et de l'intelligence artificielle qui vise à permettre aux machines de comprendre, d'interpréter et de générer du langage humain de manière naturelle.

Le TLN comprend une variété de tâches qui permettent aux ordinateurs de traiter et d'analyser le langage humain. Il s'agit notamment de la compréhension du langage, de la génération de textes, de la reconnaissance vocale, de la classification de textes et de bien d'autres choses encore.

L'objectif principal du TLN est de permettre aux machines de comprendre le langage humain et d'interagir avec lui de manière significative. Cela peut être utilisé dans une variété d'applications, telles que les assistants virtuels, les chabots, les systèmes de traduction, l'analyse de données

textuelles, la recherche d'informations, la surveillance des médias sociaux et beaucoup d'autres domaines où le langage humain est impliqué.

## 2. Algorithme choisi pour ce problème

L'algorithme qui est bien adapté et largement utilisé pour la classification de texte est *Naïve Bayes*, et cela est dû à son efficacité dans le traitement de données de grande dimension et dispersées telles que celles couramment rencontrées dans l'analyse de texte.

Naïve Bayes se distingue par son efficacité de calcul et sa capacité à s'adapter à de vastes ensembles de données, ce qui est particulièrement avantageux dans les tâches de classification de texte impliquant un grand nombre de mots ou de termes.

L'étape d'entraînement de Naïve Bayes consiste à calculer rapidement des probabilités simples basées sur l'occurrence des mots. Cela lui permet de bien fonctionner même avec de petites données d'entraînement, ce qui est bénéfique dans les tâches de classification de texte où la collecte d'un ensemble de données étiquetées conséquent peut être chronophage.

Bien que Naïve Bayes ne capture pas les relations complexes entre les mots et leur signification, il obtient souvent des performances compétitives dans les tâches de classification de texte, surtout en comparaison avec des algorithmes plus intensifs en calcul. Toutefois, il est important de noter que les performances de Naïve Bayes dépendent fortement de la qualité des données d'entraînement et de la pertinence de l'hypothèse d'indépendance dans le contexte spécifique du problème de classification de texte à résoudre.

## 3. Fonctionnement d'algorithme

Prenons l'exemple de la classification de texte où nous voulons classer les critiques de films comme "positives" ou "négatives" en utilisant Naïve Bayes.

### i. *Préparation des données*

- Recueillir un jeu de données labellisés, où chaque texte est associé à une labellisation de sentiment ("positif" ou "négatif").
- Prétraiter le texte des données.

### *ii. Extraction de features*

- Convertissez chaque texte en une représentation vectorielle de caractéristiques. Une approche courante consiste à utiliser le modèle "Bag-Of-Words", où chaque mot devient une caractéristique, et sa présence ou sa fréquence dans le texte détermine la valeur de caractéristique correspondante.

### *iii. Apprentissage*

- Calculez les probabilités a priori de chaque classe de sentiment. Cela peut être fait en comptant le nombre d'avis dans chaque classe et en divisant par le nombre total d'avis.
- Calculez les probabilités de vraisemblance de chaque caractéristique compte tenu de chaque classe de sentiment. Pour chaque classe de sentiment, comptez les occurrences de chaque mot dans les avis appartenant à cette classe et calculez la probabilité de chaque mot en fonction de la classe de sentiment.

### *iv. Classification*

- Étant donné un nouveau, passez en revue, convertissez-le en un vecteur de caractéristiques en utilisant le même processus d'extraction de caractéristiques que dans la phase de formation.
- Calculer les probabilités a posteriori de l'avis appartenant à chaque classe de sentiment en utilisant le théorème de Bayes et l'hypothèse d'indépendance.
- Pour chaque classe de sentiment, calculez  $P(\text{caractéristiques} \mid \text{sentiment})$  en multipliant les probabilités des caractéristiques individuelles compte tenu de la classe de sentiment.
- Sélectionnez la classe de sentiment avec la probabilité a posteriori la plus élevée comme sentiment prédit pour l'avis (par exemple, "positif" ou "négatif").

### *v. Evaluation*

- Comparez les étiquettes de sentiment prédites avec les véritables étiquettes d'un ensemble de données de test distinct pour évaluer la précision ou d'autres mesures de performance du classificateur Naïve Bayes.
- Ajustez le modèle ou les techniques de prétraitement si nécessaire et répétez le processus jusqu'à l'obtention de résultats satisfaisants.

## 4. Prétraitement de texte

Le prétraitement des données est le processus de nettoyage et de préparation des données textuelles pour la classification. Elle a lieu avant l'étape de vectorisation.

L'objectif principal de ce processus est de se débarrasser des informations indésirables du texte telles que la ponctuation, les stopwords (ces mots qui ne fournissent aucune information utile pour décider dans quelle catégorie un texte doit être classé), les mots alphanumériques...etc. De plus, ce processus est tellement nécessaire car il permet d'obtenir des scores élevés et d'améliorer les performances du modèle.

Voici les étapes de prétraitement de texte :

### *i. Normalisation*

- Suppression des balises d'HTML.
- Suppression des apostrophe. (par exemple : "j'ai" devient "j ai")
- Suppression des mots alphanumérique. (par exemple : " J9h2K6dL8s ")
- Suppression des accents.
- Suppression des caractères spéciaux. (par exemple : " @ & " ‘ [ ( ^ < > ... etc.")
- Faire rendre le texte minuscule.

Voilà un exemple :

```
Texte originale : Bonjour à vous! <p>J'espère que tout le monde est bien.<br>@amz21 #twitter</p> <test de fonction de normalisation>
```

```
Texte normalisé : bonjour a vous j espere que tout le monde est twitter
```

### *ii. Tokenisation*

La tokenisation est le processus de décomposition d'un texte ou d'une séquence de caractères en unités plus petites appelées "tokens". Les tokens peuvent être des mots, des phrases ou même des caractères, selon le niveau de granularité souhaité. La tokenisation est une étape fondamentale du traitement du langage naturel et joue un rôle crucial dans diverses tâches liées au langage telles que la classification des textes, l'analyse des sentiments, la détection des langues... etc.

```
>> ['bonjour', 'a', 'vous', 'j', 'espere', 'que', 'tous', 'le',  
'monde', 'est', 'twitter']
```

### *iii. Suppression des stopwords*

Les mots vides sont des mots couramment utilisés qui sont souvent supprimés du texte lors de tâches de traitement du langage naturel telles que l'exploration de texte, la recherche d'informations et l'apprentissage automatique. Ces mots sont considérés comme ayant peu ou pas d'importance pour déterminer le sens ou le contexte d'une phrase. Prenons par exemple : « **le, la, les, de, du, des, et, ou, dans ...etc.** »

```
['bonjour', 'a', 'vous', 'j', 'espere', 'que', 'tous', 'le', 'monde',
'est', 'twitter']
>> ['bonjour', 'espere', 'monde', 'twitter']
```

### *iv. Stemming / Lemmatisation*

Le *Stemming* et la *Lemmatisation* sont des techniques utilisées dans le traitement du langage naturel pour réduire les mots à leur forme de base ou racine. Les deux méthodes visent à normaliser les mots et à regrouper les différentes formes infléchies d'un même mot. Cependant, elles diffèrent dans leur approche et leur niveau de normalisation.

Le *Stemming* est un processus qui consiste à réduire les mots à leur forme de base ou racine en supprimant les suffixes ou les préfixes. Il s'agit d'appliquer des approches heuristiques ou basées sur des règles pour supprimer les affixes. La racine obtenue n'est pas toujours un mot valide dans la langue, mais elle permet une indexation et une recherche efficaces des termes apparentés. Le *Stemming* tend à être plus rapide et plus simple que la lemmatisation, mais il ne produit pas toujours des résultats exacts.

La *Lemmatisation* vise à déterminer la forme de base ou la forme dictionnaire d'un mot, connue sous le nom de lemme. Elle prend en compte le contexte et la partie du discours (Part-Of-Speech) du mot pour produire un lemme valide et significatif. La *Lemmatisation* implique généralement l'utilisation de dictionnaires et d'analyses morphologiques pour convertir les mots en leur forme de base. Elle offre des résultats plus précis que le *Stemming*, mais peut être lourdement coûteuse en termes de calculs.

```
>> ['bonjour', 'esper', 'mond', 'twitt']
```

## VI. Conclusion

Le traitement du langage naturel (TLN) est une branche de l'intelligence artificielle qui se concentre sur l'interaction entre les machines et l'être humain avec notre langage. En travaillant en arrière-plan, le TLN vise à améliorer les outils que nous utilisons quotidiennement, tels que les chatbots, les correcteurs d'orthographe et les traducteurs. En combinant le TLN avec des algorithmes d'apprentissage automatique, nous pouvons créer des systèmes qui apprennent à accomplir des tâches de manière autonome et qui s'améliorent avec l'expérience.

# *Chapitre III*

---

---

## *Conception de Projet*

---



---

### Contenu de chapitre :

- Language de modélisation
  - Diagrammes de comportement
  - Architecture générale
- 



# Chapitre III : Conception de Projet

## I. Introduction

La phase de conception d'un projet informatique est très importante, car elle nous permet de comprendre l'idée du programme, son fonctionnement et ses différentes fonctionnalités.

Ce chapitre se concentrera à la présentation du diagramme des cas d'utilisation, du diagramme de séquence et de l'architecture générale de l'application, qu'il s'agisse d'un analyseur de sentiments ou d'un détecteur de langage, afin de comprendre le mécanisme du programme.

## II. Language de modélisation UML

UML (*Unified Modeling Language*) est un langage de modélisation à usage général destiné à fournir un moyen standard de visualiser la conception d'un système. Il fournit une notation standard pour de nombreux types de diagrammes qui peuvent être grossièrement divisés en trois groupes principaux : les diagrammes de comportement, les diagrammes d'interaction et les diagrammes de structure.

- **Les diagrammes de structure** représentent les aspects statiques du système. Ils mettent l'accent sur les éléments qui doivent être présents dans le système modélisé.
- **Les diagrammes de comportement** représentent les aspects dynamiques d'un système et se concentrent sur ce qui doit se passer dans le système modélisé.
- **Les diagrammes d'interaction**, un sous-ensemble des diagrammes de comportement, mettent l'accent sur le flux de contrôle et de données entre les éléments du système modélisé.



## III. Diagramme de cas d'utilisations

Un diagramme de cas d'utilisation est une représentation graphique des interactions possibles entre un utilisateur et un système. Il illustre différents cas d'utilisation et différents types

d'utilisateurs du système et sera souvent accompagné d'autres types de diagrammes. Les cas d'utilisation sont représentés par des cercles ou des ellipses. Les acteurs sont souvent représentés par des bâtons.

Les utilisateurs du système sont représentés par **des acteurs**. Chaque acteur joue un rôle spécifique dans le système, appelé cas d'utilisation. Un même cas d'utilisation peut être confié à plusieurs acteurs. Un acteur peut être *une personne*, comme un client, ou *un ordinateur*, comme un système de base de données ou un serveur.

Dans la figure ci-dessous, le seul acteur qui existe ici est l'utilisateur. C'est-il qui va sélectionner quel opération (Analyseur de sentiment/Détection de langue) puis, il va taper un texte ensuite, le modèle fait son travail et renvoie le résultat final, soit la prédiction de polarité, soit la langue détectée. Enfin, dans l'analyseur de sentiments, deux résultats existent, c'est soit le texte saisi est positif soit négatif. Par contre au DéTECTEUR de langue, le seul résultat qui existe est la langue détectée.

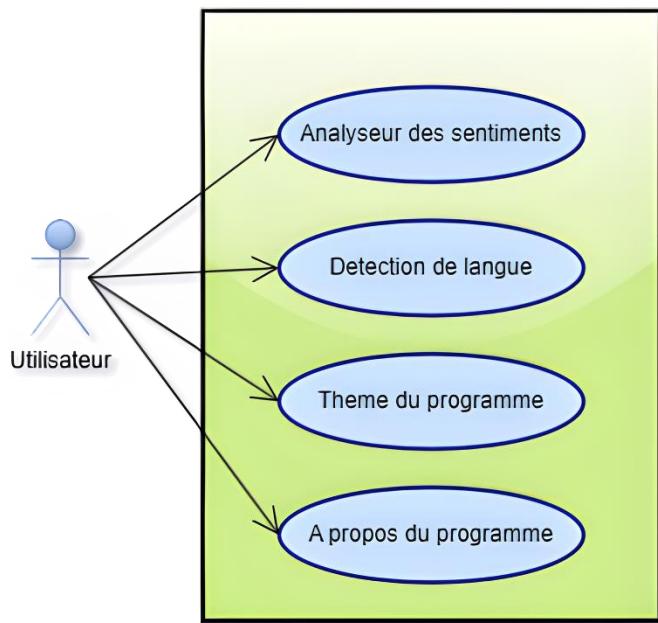


Figure 13 : Diagramme de cas d'utilisations

#### IV. Diagramme de séquence

Un diagramme de séquence est une forme de diagramme d'interaction qui illustre la manière dont plusieurs objets fonctionnent et dans quel ordre. Ces diagrammes de séquence présentent graphiquement les interactions entre les acteurs et le système, en suivant une chronologie précise.

Ils visualisent les échanges entre les objets dans le cadre d'un scenario associé à un diagramme de cas d'utilisation.

## 1. Diagramme de séquence d'Analyseur de Sentiments

Dans la figure ci-dessous, l'utilisateur choisit l'*Analyseur de sentiments* ce qui entraîne au programme de charger le modèle nécessaire pour l'Analyse de Sentiments que j'ai construit et l'appris. Puis, l'utilisateur doit taper le texte dont il veut prédire sa polarité. A ce moment-là, le programme charge la fonction de *nettoyage de texte* à partir de son paquet de fichiers pour effectuer un nettoyage pour le texte saisi. Ensuite, le programme transmet le texte nettoyé au modèle qui va le vectoriser, et qui va prédire la valeur et renvoie le résultat au programme. Enfin, le programme vérifie s'il est positif ou négatif à l'aide de l'instruction "*if-else*" et affiche la prédiction de polarité pour l'utilisateur.

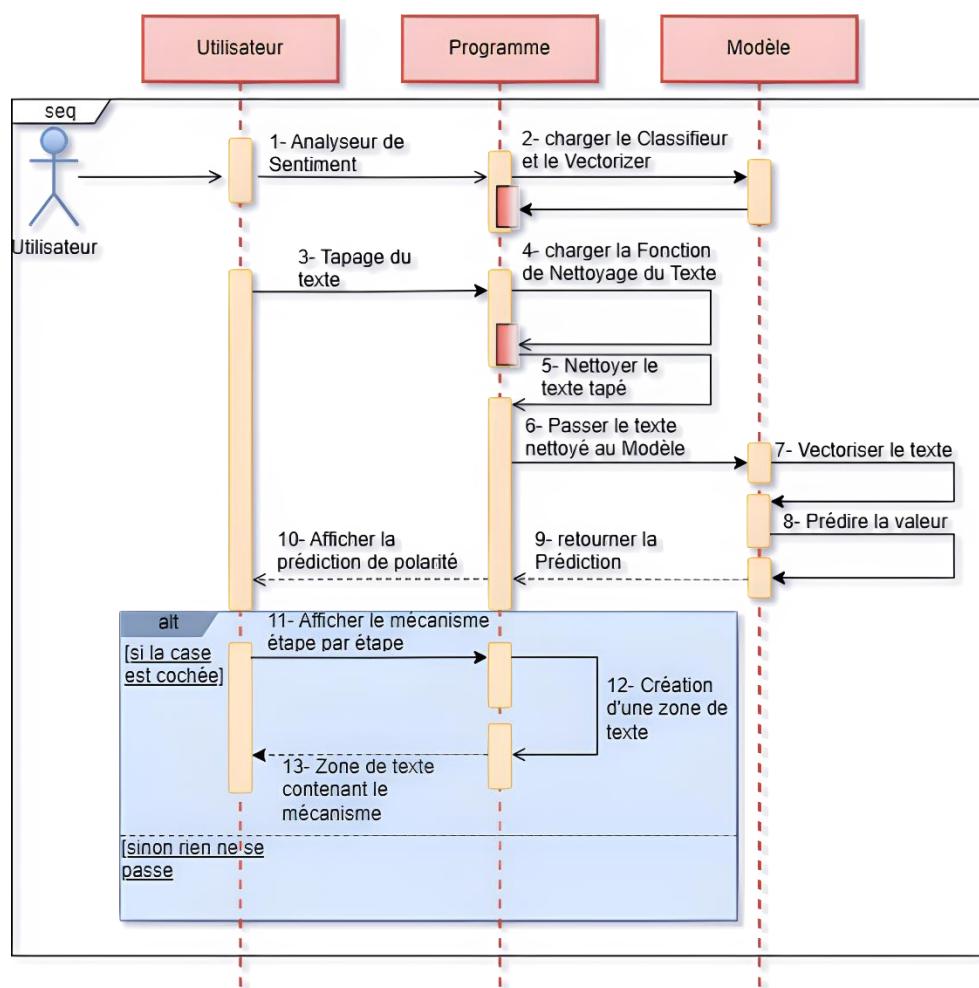


Figure 14 : Diagramme de séquence pour "Sentiment Analyzer"

## 2. Diagramme de séquence de **Détecteur de langue**

Dans la figure ci-dessous, l'utilisateur choisit le **Détecteur de langue** ce qui entraîne au programme de charger le modèle nécessaire pour le Détecteur de langue que j'ai construit et l'appris séparément de l'autre modèle. Puis, l'utilisateur doit taper le texte dont il veut détecter sa langue. A ce moment-là, le programme charge la fonction de *nettoyage de texte* à partir de son paquet de fichiers pour effectuer un nettoyage pour le texte saisi. Ensuite, le programme transmet le texte nettoyé au modèle qui va le vectoriser, et qui va prédire la langue de chaque mot puis détecter la langue du texte fourni en calculant la fréquence des mots couramment utilisés dans le texte donné et renvoie le résultat au programme. Enfin, le programme vérifie la langue détectée et l'affiche pour l'utilisateur.

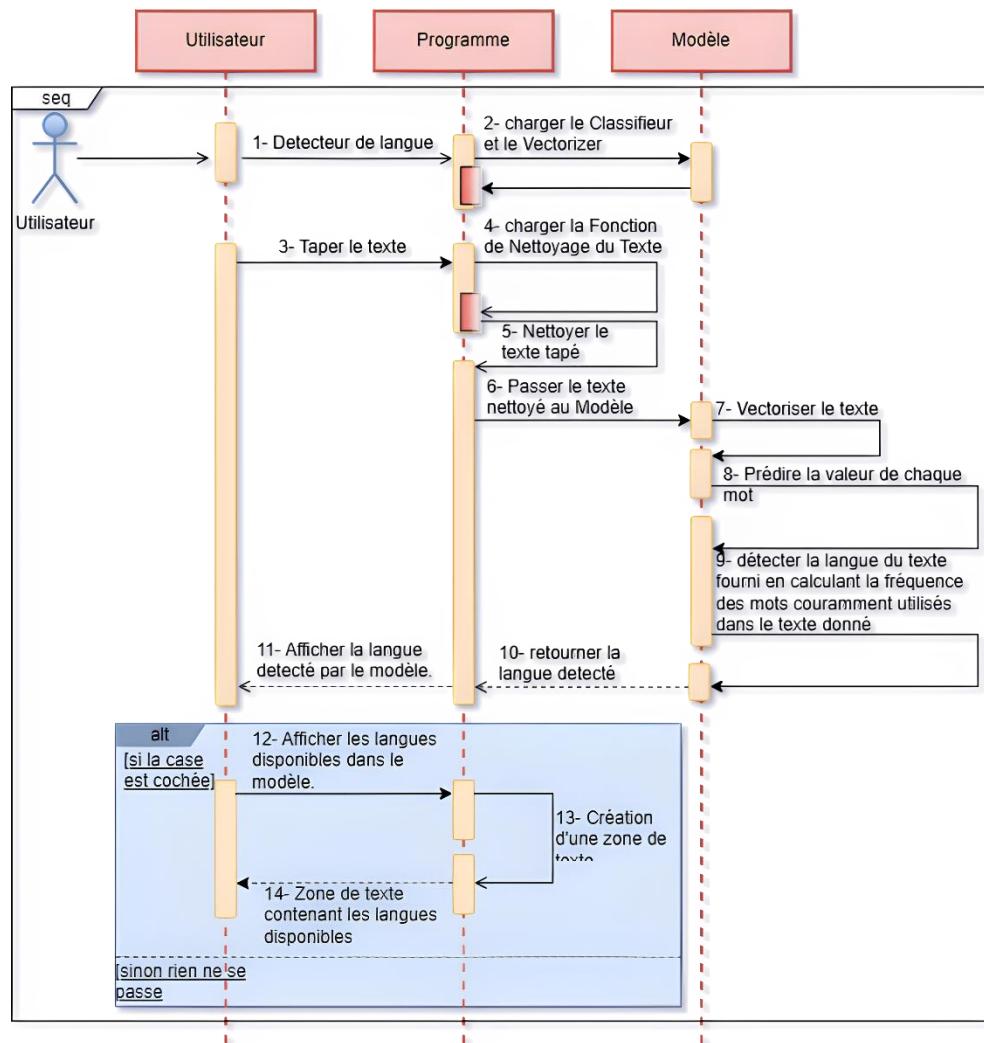


Figure 15 : Diagramme de séquence pour "Language Detecter"

Dans les deux diagrammes, on remarque que l'instruction "if-else" est identique car c'est une case à cocher. Si oui, une zone de texte apparaît.

## V. Architecture globale d'application

L'architecture d'une application décrit les modèles et les techniques utilisés pour concevoir et développer l'application. Elle fournit une feuille de route et les meilleures pratiques à suivre pour bien structurer l'application.

La figure 4 présente l'architecture générale d'*Analyseur de Sentiments* :

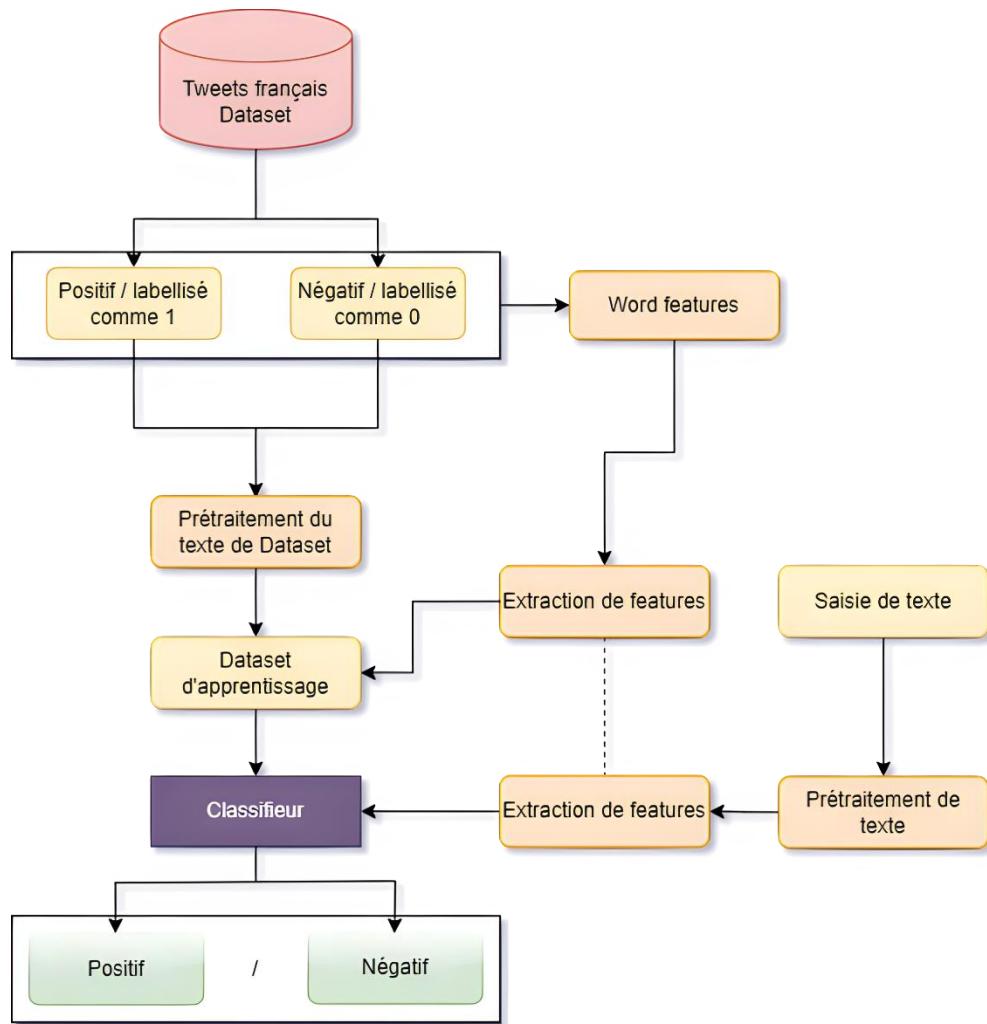


Figure 16 : L'architecture générale de "Sentiment Analyzer"

La figure 5 présente l'architecture générale de **Détecteur de langue** :

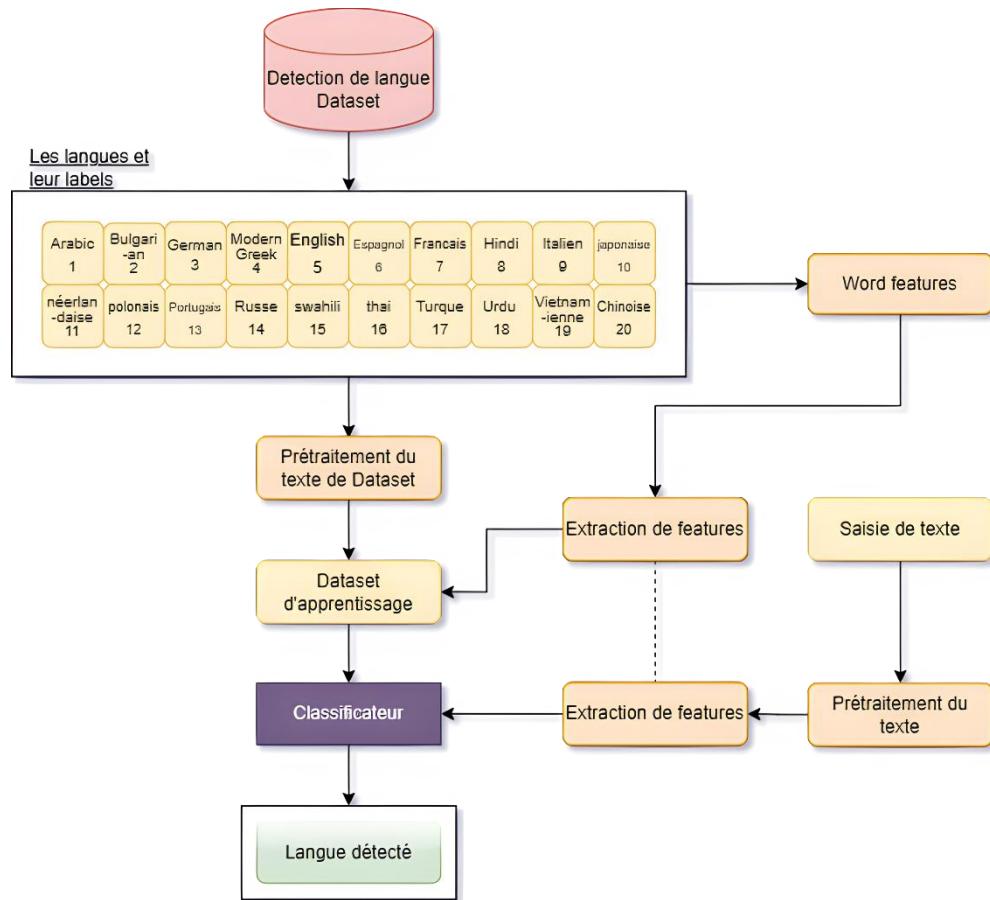


Figure 17 : L'architecture générale de "Language Detecter"

## VI. Conclusion

Dans ce chapitre, j'ai décrit les différents diagrammes pour expliquer le mécanisme de la fonction d'application. Nous pouvons maintenant procéder à la mise en œuvre de ce programme dans le chapitre suivant.

# *Chapitre IV*

---

---

## *Implémentation et Réalisation*

---



---

### Contenu de chapitre :

- Outils de développement
  - Bibliothèques utilisées
  - Implémentation et Démonstration
- 



# Chapitre IV : Implémentation et Réalisation

## I. Introduction

Dans ce chapitre, nous découvrirons les outils, les bibliothèques et les algorithmes utilisés dans le développement d'applications. De plus, nous aurons les principales interfaces du programme et enfin une démonstration.

## II. Outils de développement

Les outils essentiellement utilisés dans cette application sont les suivants :

1. Langage Python



Figure 18 : Logo de Python

*Python* est un langage de programmation open source créé par le programmeur « *Guido van Rossum* » en 1991. Son nom était tiré de l'émission « *Morty Python's Flying Circus* ».

Il s'agit d'un langage de programmation interprété et qu'il ne nécessite pas de compilation pour son fonctionnement. Il prend en charge un large éventail de paradigmes de programmation, notamment la programmation structurée, la programmation orientée objet et la programmation fonctionnelle. En raison de sa vaste bibliothèque standard, il est souvent considéré comme un langage à "piles incluses".

En tant que langage de programmation de haut niveau, *Python* permet aux programmeurs de se concentrer sur ce qu'ils font, et non sur la façon dont ils le font. Par conséquent, il prend moins de temps pour écrire un programme que dans n'importe quel autre langage. Il dispose d'un typage dynamique fort, d'une gestion automatique de la mémoire avec récupération de place et d'un système de gestion des exceptions, ce qui en fait un langage idéal pour les débutants. Il est largement utilisé dans *DATA SCIENCE*.

## 2. Installation de Python

D'abord, on visite le site de *Python* officiel "<https://www.python.org/downloads/>" et cliquons sur le bouton *Download Python 3.11.3* :



Figure 19 : Etape I d'installation de Python

Une fois le programme d'installation est téléchargé, on va l'exécuter :

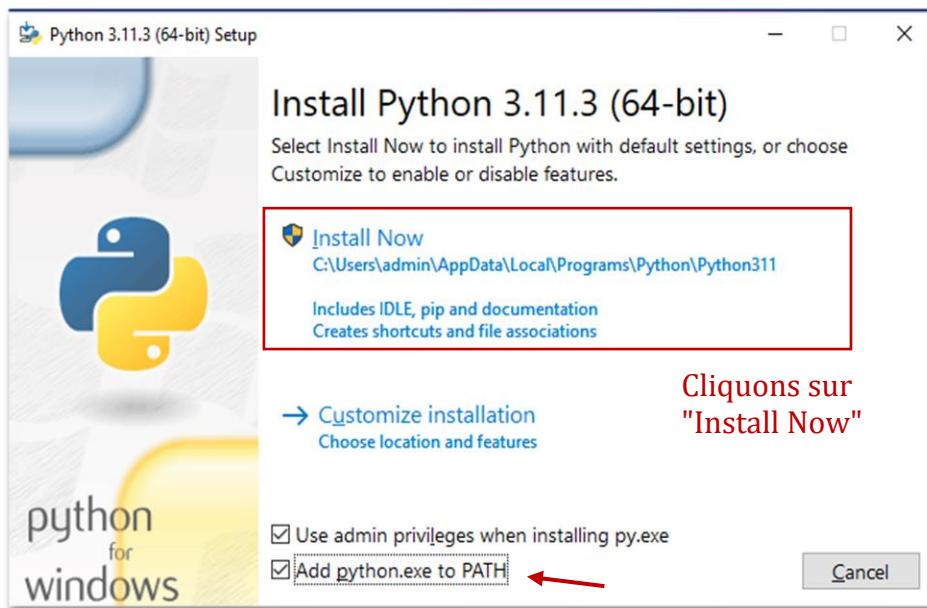


Figure 20 : Etape II d'installation de Python

Ensuite, on attend que l'installation se termine.

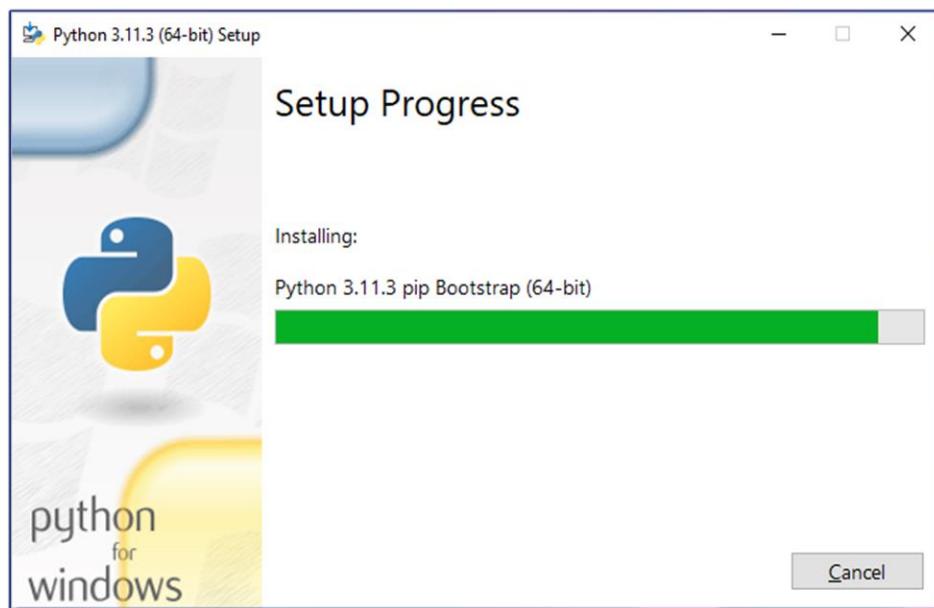


Figure 21 : Etape III d'installation de Python

Enfin, on obtient cette fenêtre :

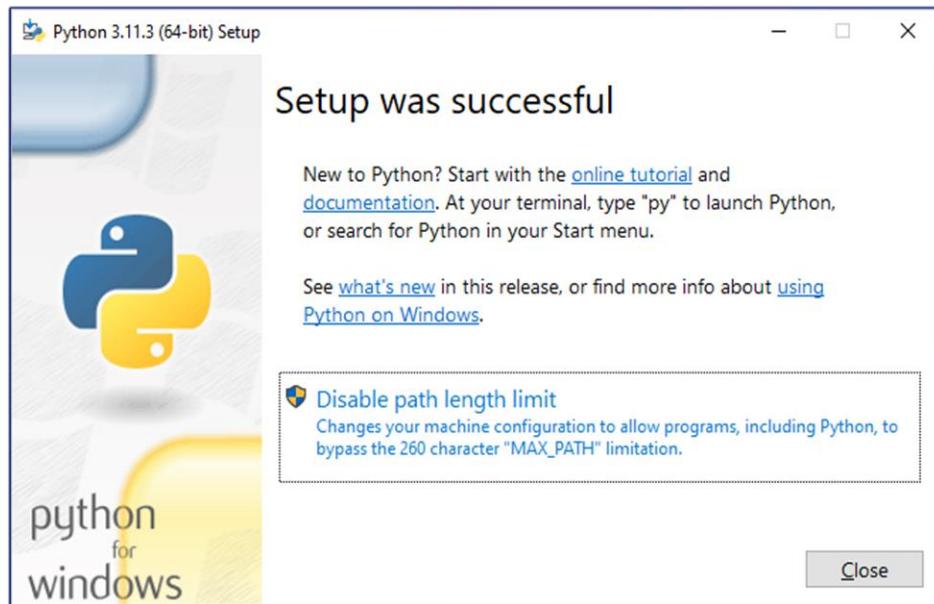


Figure 22 : Etape IV d'installation de Python

### 3. Visual Studio Code

*Visual Studio Code* est un environnement de développement intégré (IDE) qui est riche des utiles fonctionnalités. On peut l'utiliser pour modifier, débugger et écrire de code.

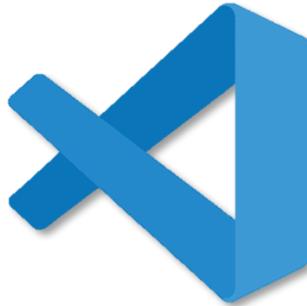


Figure 23 : Logo de VS Code

### 4. Installation de VS Code

L'installation de vs code est très simple. D'abord, on télécharge le fichier d'installation d'après le site officiel "<https://code.visualstudio.com/>" et cliquons sur *Download for Windows* :

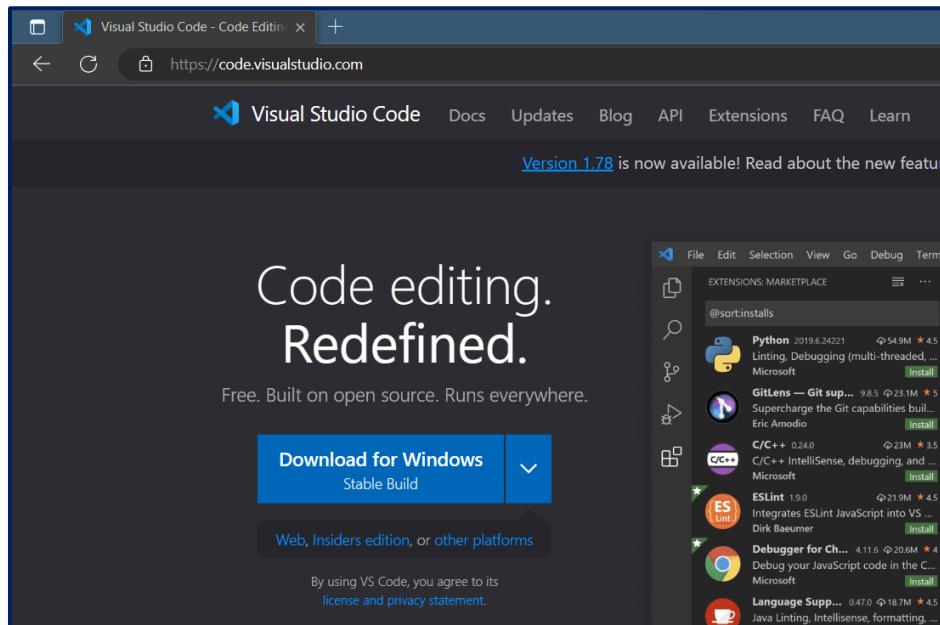


Figure 24 : Etape I d'installation de VS Code

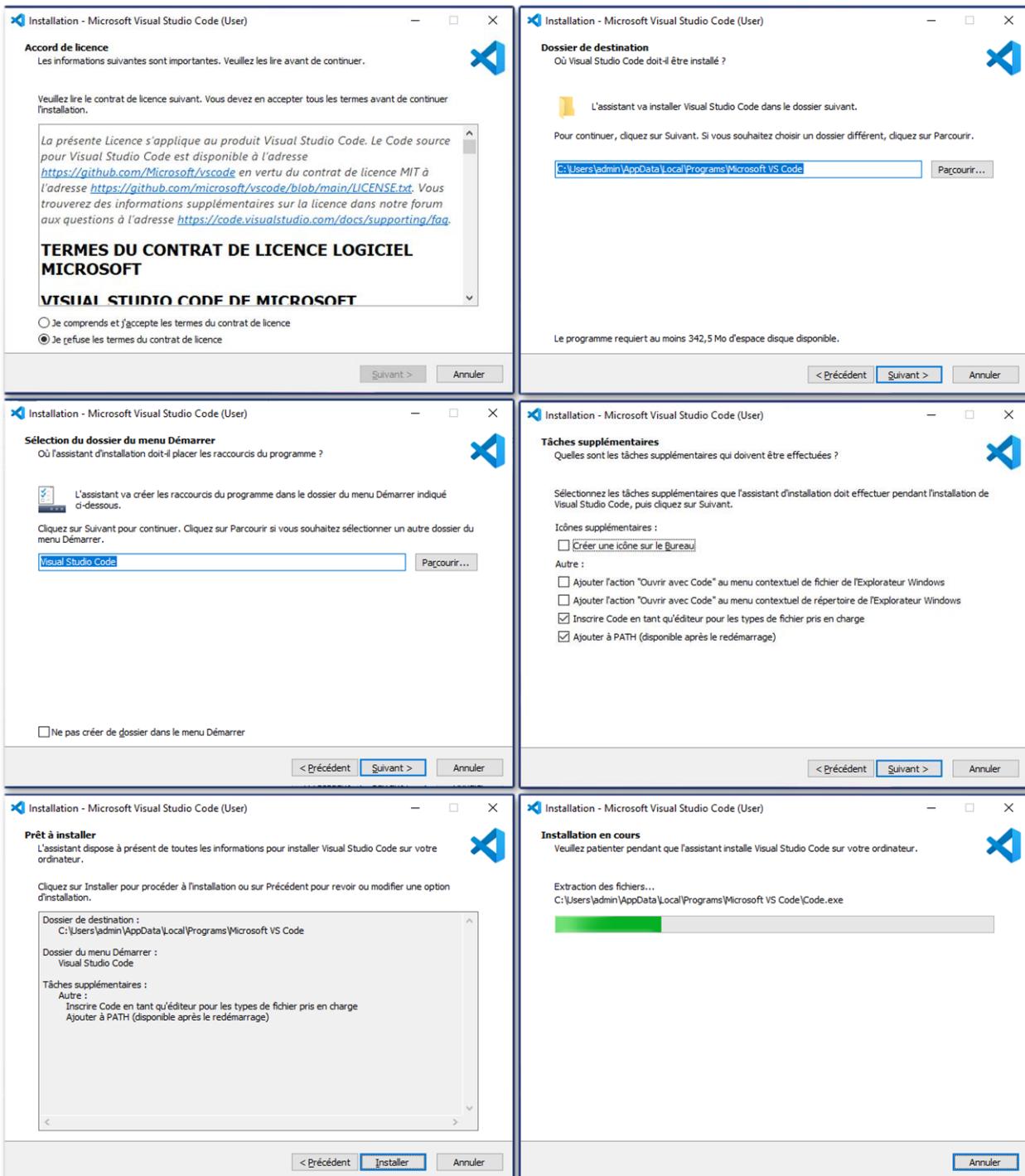


Figure 25 : Etapes d'installation de VS Code

### III. Bibliothèques utilisées :

#### 1. Pandas

*Pandas* est une bibliothèque de manipulation de données open-source pour le langage de programmation Python. Elle fournit un ensemble d'outils puissants pour travailler avec des données structurées, telles que des données tabulaires (par exemple, dans un fichier CSV ou Excel) ou des données de séries temporelles.



#### 2. NumPy

*NumPy* est une puissante bibliothèque Python qui prend en charge les tableaux et matrices multidimensionnels de grande taille, ainsi qu'une variété de fonctions mathématiques de haut niveau permettant d'opérer sur ces tableaux.



#### 3. Scikit-Learn (sklearn)

*Scikit-learn* est une bibliothèque d'apprentissage automatique à code source ouvert pour le langage de programmation Python. Elle fournit une large gamme d'outils et d'algorithmes pour diverses tâches d'apprentissage automatique, notamment la classification, la régression, le regroupement et la réduction de la dimensionnalité.



#### 4. NLTK (Natural Language Tool Kit)

*NLTK* est une bibliothèque open-source populaire pour le traitement du langage naturel (NLP) en Python.

#### 5. Custom Tkinter & Tkinter

*Tkinter* est une boîte à outils standard d'interface utilisateur graphique (GUI) pour Python. Elle fournit un ensemble de composants graphiques, tels que des boutons, des menus et des zones de texte, qui peuvent être utilisés pour créer des applications de bureau avec une interface utilisateur graphique.

*CustomTkinter* est une bibliothèque d'interface utilisateur python basée sur Tkinter, qui fournit de nouveaux widgets modernes et entièrement personnalisables. Ils sont créés et utilisés comme des widgets Tkinter normaux et peuvent également être utilisés en combinaison avec des éléments Tkinter normaux.



#### 6. Joblib

*Joblib* est une bibliothèque Python qui fournit des outils pour enregistrer et charger votre modèle personnalisé.



#### 7. BS4 (Beautiful Soup 4)

*BS4 (Beautiful Soup 4)* est une bibliothèque Python pour le web scraping et l'analyse de documents HTML et XML. Elle fournit une interface simple et facile à utiliser pour extraire des données de pages web et d'autres documents structurés.

#### 8. Re

La bibliothèque *re* est une bibliothèque Python intégrée utilisée pour travailler avec des expressions régulières. Les expressions régulières sont un outil puissant et flexible pour la

recherche et la manipulation de texte, vous permettant de faire correspondre des motifs dans des chaînes de texte et d'effectuer diverses opérations sur celles-ci.

## 9. Unidecode

La bibliothèque *Unidecode* est une bibliothèque Python utilisée pour translittérer du texte Unicode en texte ASCII. Unicode est un système de codage de caractères qui prend en charge une large gamme de caractères provenant de différents systèmes d'écriture, alors que l'ASCII (American Standard Code for Information Interchange) est un système de codage de caractères plus simple qui ne prend en charge que les caractères latins de base.

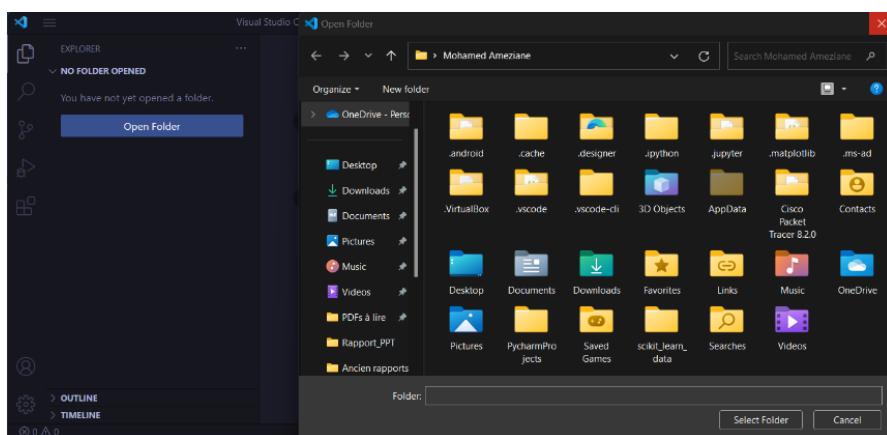
## IV. Implémentation

### 1. Création d'un projet python

Après avoir installé Python et un IDE (dans notre cas *VS Code*), il faut maintenant installer les bibliothèques nécessaires montrées dans [la partie précédente](#). Voici les étapes à suivre pour les installer en utilisant **PIP** (*Package Installer for Python*) :

- Localisez votre chemin de Scripts de Python. Dans mon cas c'est "  
`C:\Users\moham\AppData\Local\Programs\Python\Python311\Scripts`"
- En utilisant une première fois `cd` pour vous rendre au point de départ de votre disque dur, puis une deuxième fois pour changer la répertoire  
`cd C:\Users\moham\AppData\Local\Programs\Python\Python311\Scripts`
- Tapez "`pip install package_name`"

Maintenant, créons/chargeons le dossier dans lequel le projet se déroulera. Dans *VS Code*, on clique sur Open Folder puis on sélectionne le dossier approprié.



## 2. Acquisition et préparation de données

Les datasets utilisées dans ce projet peuvent être téléchargées sur [Kaggle](#) pour l'**Analyseur de Sentiments** et sur [Huggingface](#) pour le **Détecteur de Langues**. Vous devez télécharger ces datasets pour le préparer avec moi.

### i. *Chargement de Dataset*

Cette étape consiste à importer la Dataset, et comme nous le savons, Pandas est celle utilisée pour la manipulation des Datasets. On doit l'importer d'abord puis charger la Dataset à l'aide de la méthode `read_csv()` et la stocker dans une variable.

```
import pandas as pd  
df = pd.read_csv("datas/french_tweets")
```

On peut maintenant inspecter la Dataset et effectuer un nettoyage comme par exemple la suppression des valeurs nulles et les doublons ou vérifiant la taille...etc.

```
df.shape #détermination de dimension de dataset(lignes,colonnes)  
df.columns # Index(['label', 'text'], dtype='object')  
df.info() # utilisée pour obtenir un résumé, y compris l'index, le type  
et les types de colonnes, les valeurs non nulles et l'utilisation de la  
mémoire.  
df['label'].value_counts() # compter les occurrences  
df.isna().sum() #compter les occurrences des valeurs nulles  
df.duplicated().sum()#compte les occurrences des valeurs doublons
```

Au cas où la Dataset n'est pas équilibrée, on peut utiliser la méthode ci-dessous pour la rendre équilibrée.

```
df_grouped_by = df.groupby(['label'])  
df=df_grouped_by.apply(lambda  
    x:x.sample(df_grouped_by.size().min()).reset_index(drop=True))  
df = df.droplevel(['label'])
```

Enfin, on sauvegarde la Dataset

```
df.to_csv("data/Input_toClean/inspected_data.csv")
```

### ii. *Normalisation*

Comme expliqué dans l'Etat de l'art, on va écrire une fonction spécifique pour la normalisation de texte,

```
from bs4 import BeautifulSoup  
import re  
from unidecode import unidecode
```

```

def normalisation(text):
    text = BeautifulSoup(text, "html.parser")
    text = re.sub("\'", " ", text).strip()
    text = re.sub("\s*\d\S*", "", text).strip()
    text = unidecode(text)
    text = re.sub('[^a-zA-Z]', ' ', text)
    text = " ".join(text.split())
    text = text.lower()
    return text

```

Cette fonction ci-dessus commence par la suppression des *balise HTML, apostrophes, mots alphanumériques, accents, caractères spéciaux et les espaces supplémentaires*, et enfin elle rende le texte minuscule.

### *iii. Tokenisation*

Ils existent beaucoup des fonctions de tokenisation pour traitement de texte, dans cette j'ai utilisé le *word\_tokenize* qui trouve dans NLTK

```

from nltk.tokenize import word_tokenize
def tokenizer(text):
    return word_tokenize(text, language='french')

```

et comme résultat, on aura un tableau contenant les mots de texte. Cela pour simplifier la suppression des stopwords.

### *iv. Suppression des stopwords*

Il faut d'abord télécharger la liste des stopwords en français qui se trouve dans le *corpus* de NLTK

```

from nltk.corpus import stopwords
french_stopwords = stopwords.words('french')
deselect_stopwords = ['n', 'ne', 'pas', 'plus', 'jamais', 'guere',
'personne', 'aucun', 'ni', 'aucune', 'rien']
for word in deselect_stopwords:
    if word in french_stopwords:
        french_stopwords.remove(word)
    else:
        continue
def stopwords_remover(tokens):
    return [word for word in tokens if word not in french_stopwords]

```

On obtient un tableau contenant les mots de texte sans les stopwords.

## v. Stemmmisation

Pour cela j'ai utilisé le *FrenchStemmer* de la *NLTK*, car il est basé sur l'algorithme de *Snowball Stemmer* qui est connu pour sa précision et son efficacité dans plusieurs langues.

```
from nltk.stem.snowball import FrenchStemmer
def stemmer(tokens):
    stemmer = FrenchStemmer()
    return [stemmer.stem(word) for word in tokens]
```

## vi. Application sur la Dataset

Dans le code ci-dessous, j'ai sélectionné la colonne de text d'après la dataset puis j'ai utilisé la méthode map() applique une fonction à chaque élément d'un tableau itératif.

```
text = df['text']
cleaned_text = text.map(text_Cleaner)
df['cleaned_text'] = cleaned_text
```

Enfin, on sauvegarde la Dataset nettoyée.

## 3. Phase d'apprentissage de la machine

Après avoir préparer les données, on passe à une importante phase c'est de l'apprentissage.

### i. Division de Dataset en celle de test et celle d'apprentissage

Avec le code ci-dessous, on va séparer la Dataset en 20% pour les tests et 80% pour l'apprentissage.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=0)
```

### ii. Vectorization (Bag-of-words)

Cette action consiste à créer la matrice d'occurrence des différents mots dans les différentes phrases. Cette matrice sert à déterminer le nombre de fois ou apparaît un mot dans le texte.

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features=5000)

x = cv.fit_transform(df['cleaned_text'].values.astype(str).tolist())
    .toarray()
y = df.iloc[:,0]
```

### *iii. Apprentissage du modèle*

Maintenant, on peut attaquer l'apprentissage de la machine et pour cela j'ai choisi l'algorithme de naïve bayes en raison de sa simplicité, de son efficacité et de sa capacité à traiter des données de haute dimension. Après avoir essayé les trois algorithmes de bayes naïve (*GaussianNB*, *MultinomialNB* et *BernoulliNB*), celui qui s'est le mieux adapté au problème de la classification de textes est **MultinomialNB**.

```
from sklearn.naive_bayes import MultinomialNB  
classifier = MultinomialNB()  
classifier.fit(x_train, y_train)
```

### *iv. Score et Performance*

Pour la vérification du modèle construit, on va utiliser les données de test qu'on a déjà préparé puis on va utiliser ces trois fonctions pour montrer le score :

- Confusion Matrix
- Accuracy Score
- Classification Report

```
from sklearn.metrics import classification_report, confusion_matrix,  
accuracy_score  
y_pred = classifier.predict(x_test)  
print(confusion_matrix(y_test, y_pred))  
print(accuracy_score(y_test, y_pred))  
print(classification_report(y_test, y_pred))
```

Enfin, le score obtenu est illustré dans la figure ci-dessous ;

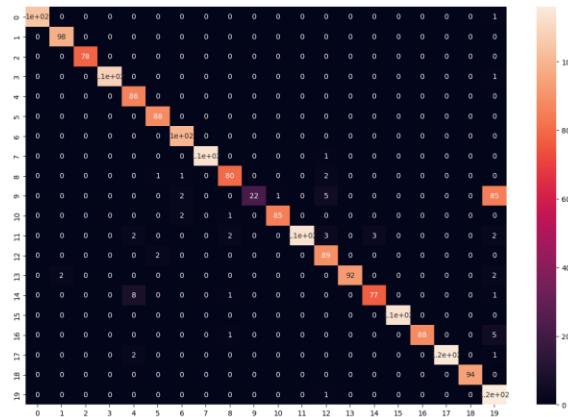
#### i. Pour le modèle d'Analyse de sentiment

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score  
  
print(confusion_matrix(y_test, y_pred),'\n-----')  
print(accuracy_score(y_test, y_pred),'\n-----')  
print(classification_report(y_test, y_pred),'\n-----')  
  
[[9886 3214]  
 [3265 9635]]  
-----  
0.7508076923076923  
-----  
 precision    recall   f1-score   support  
  
      0       0.75      0.75      0.75     13100  
      1       0.75      0.75      0.75     12900  
  
   accuracy                           0.75      26000  
 macro avg       0.75      0.75      0.75      26000  
weighted avg     0.75      0.75      0.75      26000
```

Figure 26 : Score de performance de modèle d'Analyse des sentiments

## ii. Pour le modèle de Détection des langues

	precision	recall	f1-score	support
1	1.00	0.99	1.00	105
2	0.98	1.00	0.99	98
3	1.00	1.00	1.00	78
4	1.00	0.99	1.00	109
5	0.88	1.00	0.93	86
6	0.97	1.00	0.98	88
7	0.95	1.00	0.98	100
8	1.00	0.99	1.00	114
9	0.94	0.95	0.95	84
10	1.00	0.19	0.32	115
11	0.99	0.97	0.98	88
12	1.00	0.90	0.95	125
13	0.88	0.98	0.93	91
14	1.00	0.96	0.98	96
15	0.96	0.89	0.92	87
16	1.00	1.00	1.00	113
17	1.00	0.94	0.97	94
18	1.00	0.97	0.99	118
19	1.00	1.00	1.00	94
20	0.54	0.99	0.70	117
...				
macro avg	0.95	0.94	0.93	2000
weighted avg	0.95	0.93	0.92	2000



Il faut aussi sauvegarder le modèle de classification et celle de bag of words en utilisant « *joblib* », et spécifiquement la fonction ***dump***.

## V. Interfaces Principales

Commençons par l'interface principale où l'utilisateur est placé devant deux boutons qui sont **Sentiment Analyzer** et **Language Detecter** et aussi la possibilité de changer **l'apparence du programme (Clair/Sombre/Système)** et un bouton **About** pour lire la description de programme.

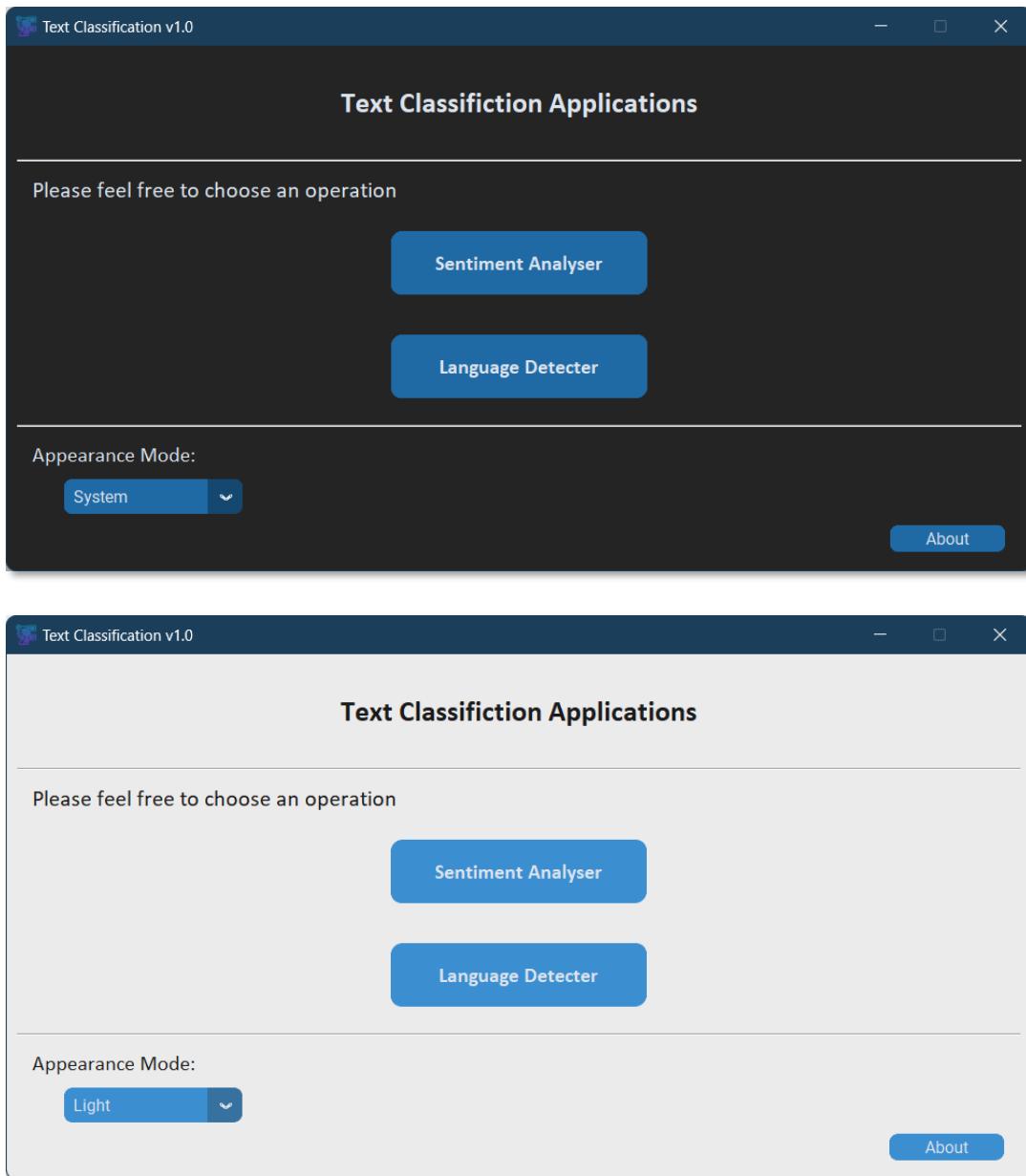


Figure 27 : Interfaces Principaux (Sombre/Clair)

Puis, on passe à l'interface de **Sentiment Analyzer** où l'utilisateur doit taper son texte puis cliquer sur le bouton pour afficher le résultat. Au cas où il souhaite voir le processus de prétraitement du texte, il suffit de cocher la case comme indiqué à l'écran.

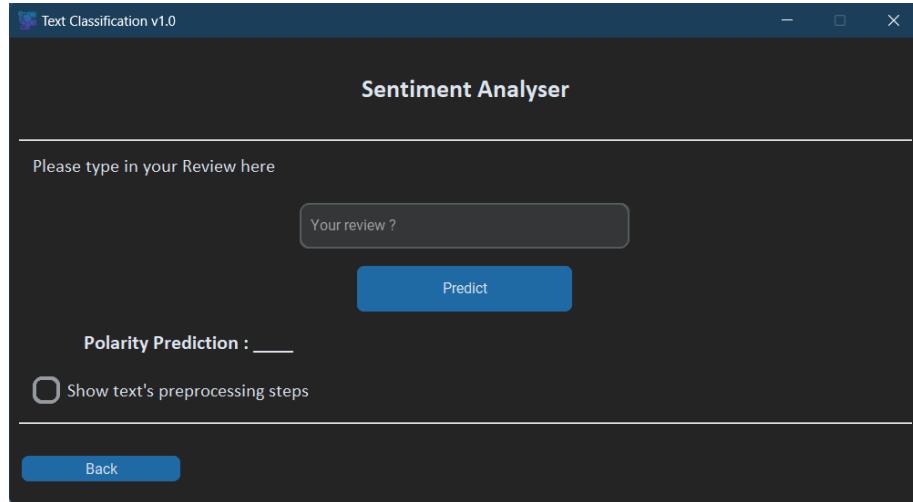


Figure 28 : Interface de "Sentiment Analyzer"

Ensuite, on passe vers l'interface du **Détecteur de Langue** où l'utilisateur doit taper le texte dans la zone de texte, puis cliquer sur le bouton pour détecter la langue dans laquelle le texte est écrit.

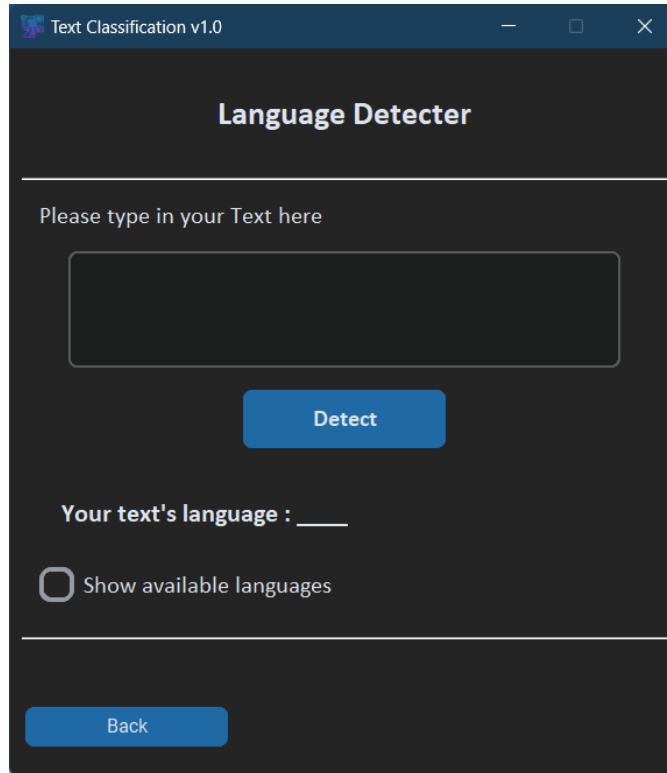


Figure 29 : Interface de "Language Detecter"

## VI. Démonstration

Testons maintenant le programme en commençant par le **Sentiment Analyzer**, donnons-lui une fois un texte positif et un texte négatif.

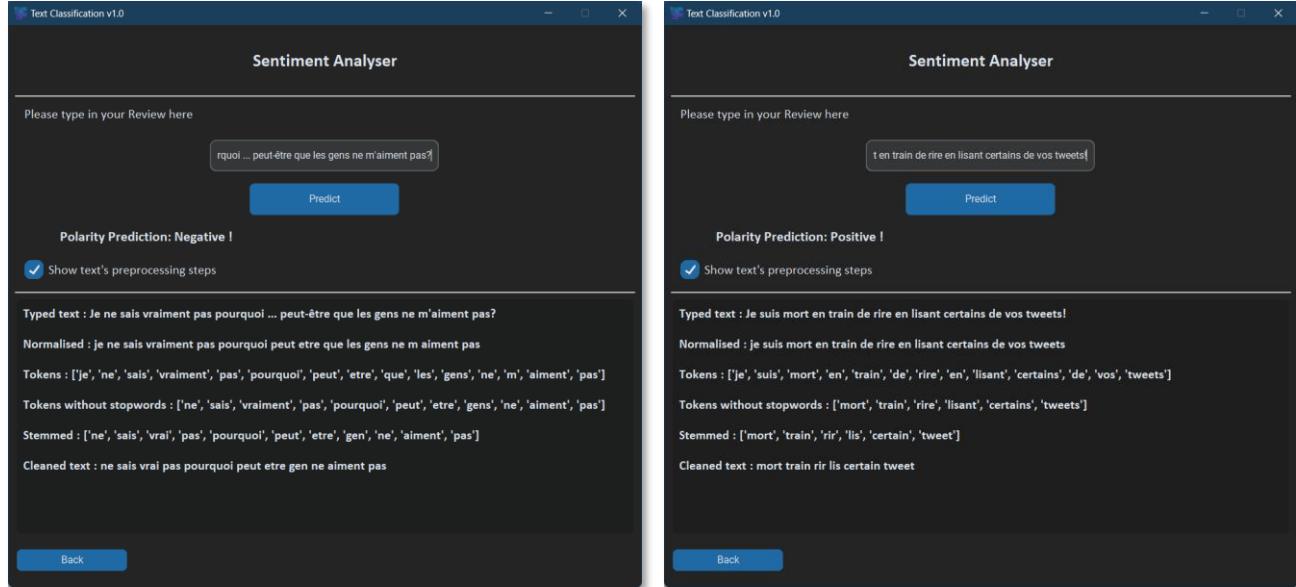
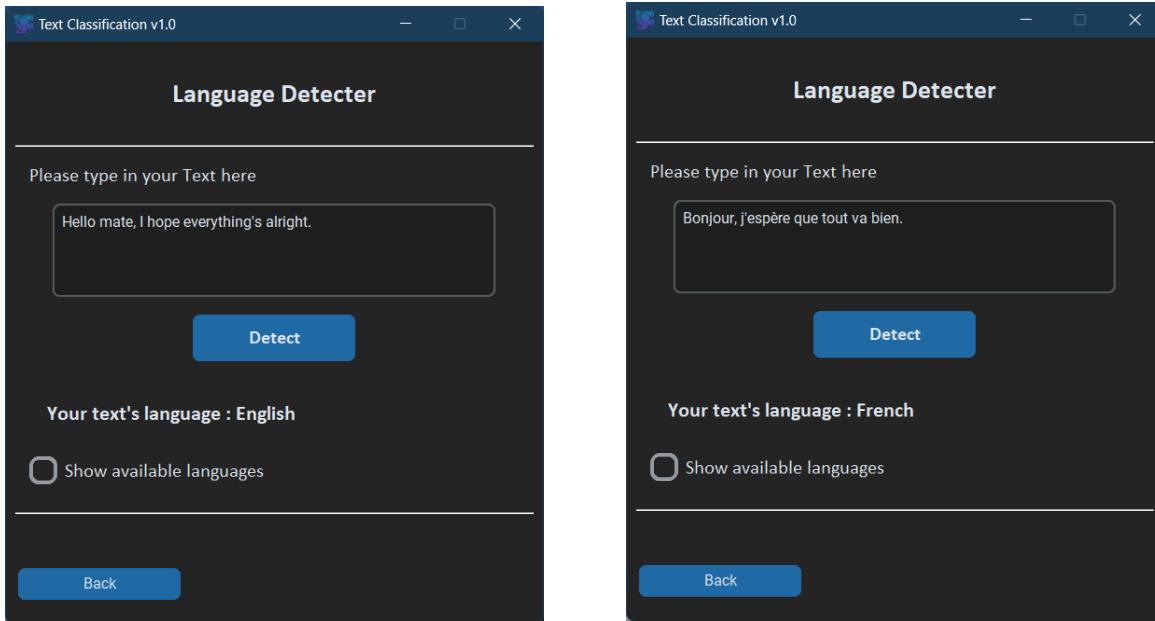


Figure 30 : Démonstration de "Sentiment Analyzer"

Ensuite, passant vers le **Détecteur de langues**, donnons-lui des textes en plusieurs langues.



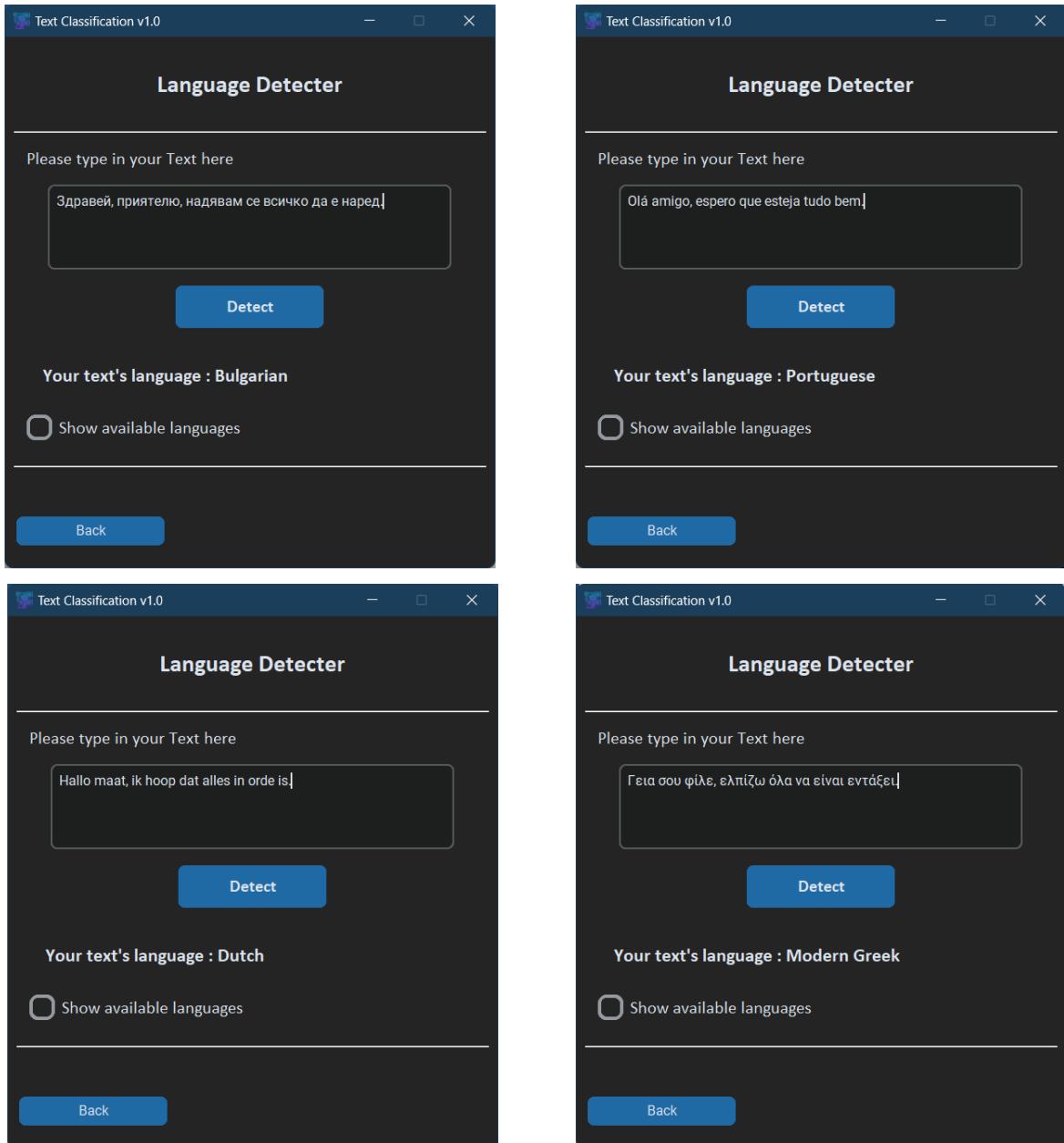


Figure 31 : Démonstration de "Language Detecter"

## VII. Conclusion

Dans ce dernier chapitre, j'ai commencé à présenter les outils, les bibliothèques utilisées dans le développement du programme et le processus d'implémentation bien détaillé afin de réaliser ce projet, en respectant la conception déjà faite dans le chapitre précédent.

## Conclusion Générale

---

Les travaux réalisés dans le cadre de cette recherche s'inscrivent dans le domaine de l'intelligence artificielle et plus particulièrement de la classification de textes. Je me suis concentré sur deux applications de la classification de texte supervisée, à savoir l'analyse des sentiments et la détection des langues.

Dans le rapport, j'ai essayé de couvrir le mécanisme de préparation des données et de leur nettoyage, puis je suis passé au processus de prétraitement du texte. Ensuite, j'ai choisi l'algorithme des bayes naïves pour l'entraîner sur mon ensemble de données nettoyé et j'ai obtenu un score de **75 %** pour le modèle d'analyse des sentiments et de **92 %** pour le modèle de détection des langues.

Il est encore possible d'améliorer ce programme en envisageant d'utiliser l'apprentissage profond ou en augmentant la taille de l'ensemble de données.

## Références

---

---

- [1] Sarkar, D. (2016). *Text Analytics with Python* (1<sup>re</sup> éd.). Apress Berkeley, CA.
- [2] VANNIEUWENHUYZE, A. (2019). *Intelligence artificielle vulgarisée - Le Machine Learning et le Deep Learning par la pratique* (RIIAVUL).
- [3] BELAININE, B. (2017). *Classification supervisée de textes courts et bruités: Application au domaine des médias sociaux* [Mémoire de fin d'étude]. <https://archipel.uqam.ca/9944/1/M15026.pdf>
- [4] RISCH, J.-C. (2017). *Enrichissement des Modèles de Classification de Textes Représentés par des Concepts* [Thèse]. <https://www.theses.fr/2017REIMS012.pdf>
- [5] Mosteghanemi, S., & Feroukhi, A. (2020). *Système de recommandation multilingue basé sur l'analyse des sentiments des opinions dans les commentaires en ligne* [Mémoire de fin d'étude]. <https://di.univ-blida.dz/jspui/bitstream/123456789/11371/1/Mosteghanemi%20Samira%20%20et%20%20Feroukhi%20Afaf.pdf>
- [6] Saker, A., Dartigues-Pallez, C., & Gaetan, R. (2020). *Classification supervisée de données pédagogiques pour la réussite dans l'enseignement supérieur* [Rapport de recherche]. <https://hal.science/hal-02486729/document>
- [7] ZIANI, A. (2018). *La recommandation via l'analyse d'opinions* [Thèse]. <https://biblio.univ-annaba.dz/wp-content/uploads/2020/01/These-Ziani-Amel.pdf>
- [8] Federchuk, M. (2022). *The Difference Between AI, ML and DL*. CENGN. <https://www.cengn.ca/information-centre/innovation/difference-between-ai-ml-and-dl/>
- [9] Roy, R. (2020). *Understanding the difference between AI, ML and DL*. <https://towardsdatascience.com/understanding-the-difference-between-ai-ml-and-dl-cceb63252a6c>
- [10] Team, D. (2018). *Deep Learning vs Machine Learning—Demystified in Simple Words*. DataFlair. <https://data-flair.training/blogs/deep-learning-vs-machine-learning/>

- [11] GeeksforGeeks. (2017). Naive Bayes Classifiers.  
<https://www.geeksforgeeks.org/naive-bayes-classifiers/>
- [12] Ray, S. (2017). Naive Bayes Classifier Explained : Applications and Practice Problems of Naive Bayes Classifier. *Analytics Vidhya*.  
<https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>
- [13] All Things Interactive. (2016). *Three Types of Learning That Artificial Intelligence (AI) Does*. <http://www.allthingsinteractive.com/new-blog/2016/12/3/three-types-of-learning-that-artificial-intelligence-ai-does>
- [14] *Supervised vs. Unsupervised Learning : What's the Difference?* (2022).  
<https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning>