Epreuve Python

Informations Générales

- ✓ L'épreuve comporte 3 missions à résoudre en **3 heures** consécutives :
  - ***Mission1 :*** *Un QCM en ligne comporte 25 questions*
  - ***Mission2 :*** *Problème1*
  - ***Mission3 :*** *Problème2*
- ✓ Chaque équipe doit créer un dossier de travail portant le nom **« SalleN°_TableN° »** qui sera utilisé pour enregistrer tous les fichiers solutions de l'épreuve.

Exemple : Salle1_Table3

Evaluation & calcul du score

- ✓ Les solutions proposées par les candidats sont évaluées par les membres du jury et notées en fonction de leur exactitude (les bugs ne sont pas tolérés) et de leur vitesse et utilisation mémoire (généralement liées à la complexité en temps et en mémoire de l'algorithme implémenté).
- ✓ Grille de répartition des scores :

Mission1	Mission2	Mission3	Bonus	(*)
30 points	30 points	30 points	10 points	-

compétences de vie : 10 points attribués **pendant l'épreuve** par les membres du jury pour évaluer les capacités des candidats à collaborer et à travailler en équipe.

Mission1 [QCM en ligne]

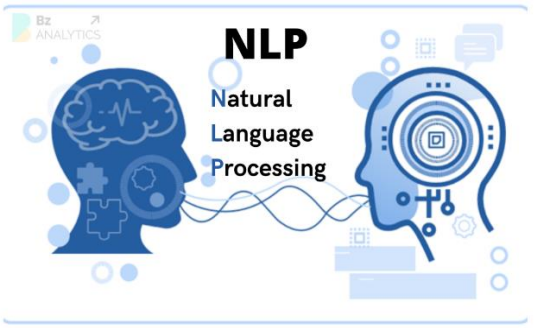
QCM sur le langage de programmation python comportant 25 questions.

Les règles d'une épreuve en ligne :

- *Durée de l'épreuve : 30 mn*
- *C'est interdit d'utiliser les smartphones,*
- *C'est interdit de naviguer entre les fenêtres ou d'utiliser des IDE (vs-code, Thonny, ect)*

Mission2 [Problème1]

Technologie du traitement du langage naturel (NLP)



L'utilisation de l'intelligence artificielle (IA) pour détecter la langue d'un texte trouve des applications dans divers domaines, notamment : Technologies de Traitement du Langage Naturel (NLP), Traduction Automatique, Analyse de Données Sociales,...

L'intégration de la détection automatique de la langue dans ces domaines permet d'améliorer l'expérience utilisateur, de personnaliser les services, et d'optimiser le traitement des données dans un contexte multilingue.

Cette approche permet de développer des modèles automatisés capables d'identifier la langue dans laquelle un texte est rédigé sans avoir besoin d'une intervention humaine directe.

Voici une approche simple pour détecter la langue d'un texte en utilisant les stop-words : Stop-Words (mot vide) est un mot non significatif figurant dans un texte caractérisant une langue.

Etape1 :

Collecte des Stop-Words : Rassemblez des listes de stop-words pour différentes langues. Il existe plusieurs bibliothèques qui fournissent déjà des listes de stop-words pour différentes langues. Exemple la bibliothèque **nltk.** Vous trouverez ci-bas comment utiliser ce module avec 4 langues '**english**', '**french**', '**spanish**'et '**german**':

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
languages = ['english', 'french', 'spanish', 'german']  # Ajoutez d'autres langues au besoin
def collecte(stop_words):
    for lang in languages:
        stop_words.append(stopwords.words(lang))
#Résultat execution:
>>> %Run 'STEM NLP.py'
    [nltk_data] Downloading package stopwords to
    [nltk_data]     C:\Users\Dell\AppData\Roaming\nltk_data...
    [nltk_data]   Package stopwords is already up-to-date!
    ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', " …
    ['au', 'aux', 'avec', 'ce', 'ces', 'dans', 'de', 'des', 'du', 'elle',  …
    ['de', 'la', 'que', 'el', 'en', 'y', 'a', 'los', 'del', 'se', 'las', ' …
    ['aber', 'alle', 'allem', 'allen', 'aller', 'alles', 'als', 'also', 'a …
```

Vous pouvez maintenant utiliser **stop_words[0]** pour les stop_words en english, **stop_words[1]** pour les stop_words en french, ect.....

Etape2 :

Tokenisation du Texte, il s'agit de :

- saisir une chaîne de caractères composée de lettres, de chiffres et d'espaces (deux mots consécutifs sont séparés par un seul espace) avec les signes de ponctuations suivants (., ;:!?).
- Ignorer tous les tokens qui sont des signes de ponctuation.
- Divisez le texte en mots (un mot est appelé token).
- Pour chaque langue, calculez le Nombre de mots du texte présents dans la liste de stop_words c'est-à-dire la fréquence de chaque mot dans le texte existant dans les listes de stop-words.

Exemple:

>>> saisie d'un texte avec respect des contraintes

saisir un texte: My Wonderful Family. I live in a house near the mountains. I have two brothers and one sister, and I was born last. My father teaches mathematics, and my mother is a nurse at a big hospital. My brothers are very smart and work hard in school. My sister is a nervous girl, but she is very kind. My grandmother also lives with us. She came from Italy when I was two years old. She has grown old, but she is still very strong. She cooks the best food!

Résultat texte après nettoyage de signe de ponctuation

My Wonderful Family I live in a house near the mountains I have two brothers and one sister and I was born last My father teaches mathematics and my mother is a nurse at a big hospital My brothers are very smart and work hard in school My sister is a nervous girl but she is very kind My grandmother also lives with us She came from Italy when I was two years old She has grown old but she is still very strong She cooks the best food

Fréquence de chaque mot pour chaque langue :

['english', 'french', 'spanish', 'german']

[33, 0, 5, 5]

Etape3 :

Score de Similarité : Calculez un score entre le texte et chaque liste de stop-words avec la formule suivante :

$$score\ de\ Similarité = \frac{\text{Nombre de mots du texte présents dans la liste de stop\_words}}{\text{Nombre Total de mots dans le texte}} \times 100$$

Exemple :

Nombre Total de mots dans le texte :

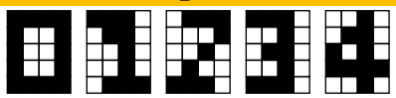
89

Score de Similarité :

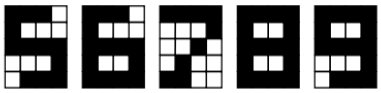
[37.08, 0.0, 5.62, 5.62]

Etape4 : Choix de la Langue : La langue avec le score de similarité le plus élevé est probablement la langue du texte. Dans l'exemple est **37.08%** qui est équivalent à **English**.

Mission3 [Problème2]



Système de Reconnaissance de Chiffres Manuscrits

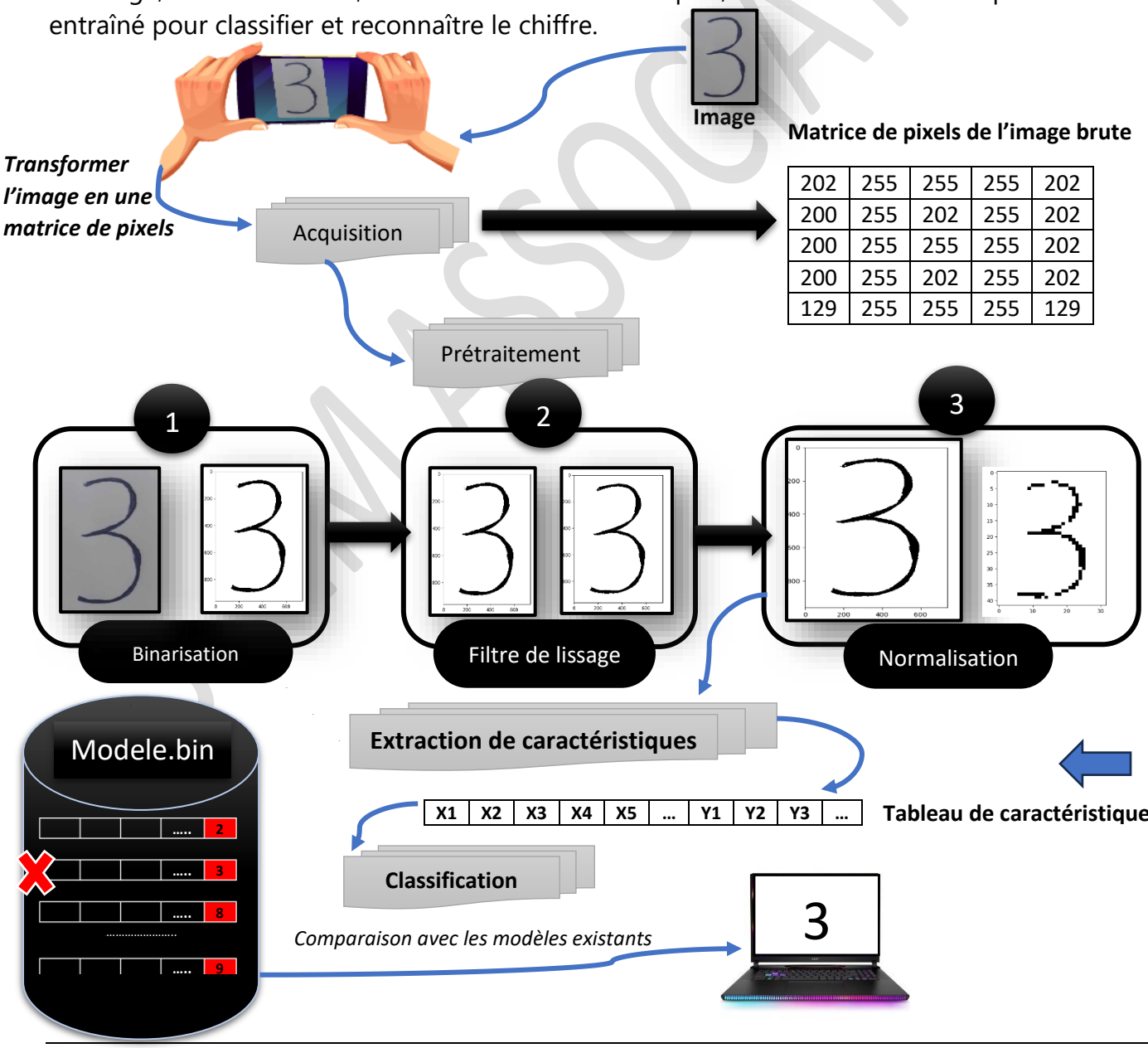


Introduction

Les Systèmes de Reconnaissance de Chiffres Manuscrits (SRCM) occupent une position cruciale au sein de l'intelligence artificielle, combinant la vision par ordinateur et l'apprentissage automatique pour interpréter et comprendre les chiffres écrits à la main. Ces systèmes sont spécialement conçus pour analyser des données provenant de diverses sources telles que des documents scannés, des formulaires manuellement remplis, et d'autres supports similaires.

Votre Mission :

Développer un script en Python permettant à l'utilisateur de charger une image contenant un unique chiffre manuscrit. Le script effectuera un traitement d'images comprenant la binarisation, le filtrage, la normalisation, l'extraction de caractéristiques, et utilisera un modèle préalablement entraîné pour classifier et reconnaître le chiffre.



Important :

- *Installer la bibliothèque :* ***PIL***
- *Toutes les ressources à utiliser dans cette mission se trouvent dans le dossier* ***ressources***
- *Le contenu de dossier* ***ressources*** *:*
  - *Le fichier image :* ***chiffre.jpg***
  - *Le fichier binaire* ***modele.bin***
  - *10 fichiers images*
  - *Un fichier texte :****code.txt***

Démarche de résolution

Tâche1 : Chargement de l'image

- *Copier l'image « chiffre.jpg » situés dans le dossier ressources dans votre dossier de travail.*
- Ouvrir l'image *« chiffre.jpg » qui* contient un chiffre manuscrit. *(Voir annexe)*
  N.B : Le résultat de l'ouverture, est un objet image contenant une grille de pixels, où chaque pixel a une position (x, y).

Tâche2 : Conversion de l'objet image en matrice.

- Déterminer les dimensions de la matrice (nombre de lignes (L) et nombre de colonnes(C)) en utilisant la propriété **size**. (voir annexe)
- Remplir une matrice M par la valeur de chaque pixel en utilisant la méthode **getpixel((x, y))** (voir annexe). Chaque élément de la matrice contient la valeur du niveau de gris du pixel de l'image. (Valeur entre 0 et 255)

Tâche3 : Prétraitement :

A) Binarisation [Conversion de l'image en une forme binaire distinguant le chiffre du fond]

La première étape de prétraitement est la binarisation de l'image qui consiste à mettre à 1 tous les pixels ayant un niveau de gris inférieur à une certaine valeur seuil et à la valeur zéro les pixels ayant une valeur supérieure.

Tout pixel au-dessus d'un seuil=100 sera considéré **« noir » et «blanc»** dans le cas contraire.

$$M[i,j] = \begin{cases} 1, M[i,j] < 100 \\ 0, M[i,j] \geq 100 \end{cases}$$

B) Filtrage : Application d'un filtre pour améliorer la qualité de l'image. Principe de l'algorithme filtre :

- Parcourir la matrice M de l'image en **excluant (sauf)** les bords.
- Pour chaque pixel, calculer la moyenne des valeurs des pixels voisins.
- La nouvelle valeur du pixel dans la matrice filtrée est alors déterminée par cette moyenne.
- Le processus est répété pour chaque pixel intérieur, produisant ainsi une version lissée de l'image originale.

Exemple d'application de l'algorithme de filtre sur le pixel (1,1)=$\frac{1+1+1+1+0+1+1+0+1}{9}$=0

1	1	1	0	1
0	1	1	0	0
1	1	0	1	0
0	1	1	1	0
1	0	0	0	1

Pixel (1,1)

1	1	1	0	1
0	0	1	0	0
1	1	0	1	0
0	1	1	1	0
1	0	0	0	1

C) Normalisation : Ramener la matrice M de l'image à une taille (42 lignes x 32 colonnes)

La taille d'un chiffre peut varier d'une écriture à l'autre, ce qui peut causer une instabilité des paramètres.

On vous propose une technique qui permet de ramener l'image à la taille (42*32) pixels.

Soit **M** de taille L×C qu'on souhaite ramener à une **M1** de taille 42×32.

Technique de normalisation :

Pour chaque pixel (i, j) de la Matrice2, le pixel correspondant dans la Matrice1 est donné par les coordonnées suivantes : ([(i * L) // 42], [(j * C) // 32]).

Tâche3 : Extraction de Caractéristiques : [Identification des propriétés distinctives d'un chiffre manuscrit dans l'image]

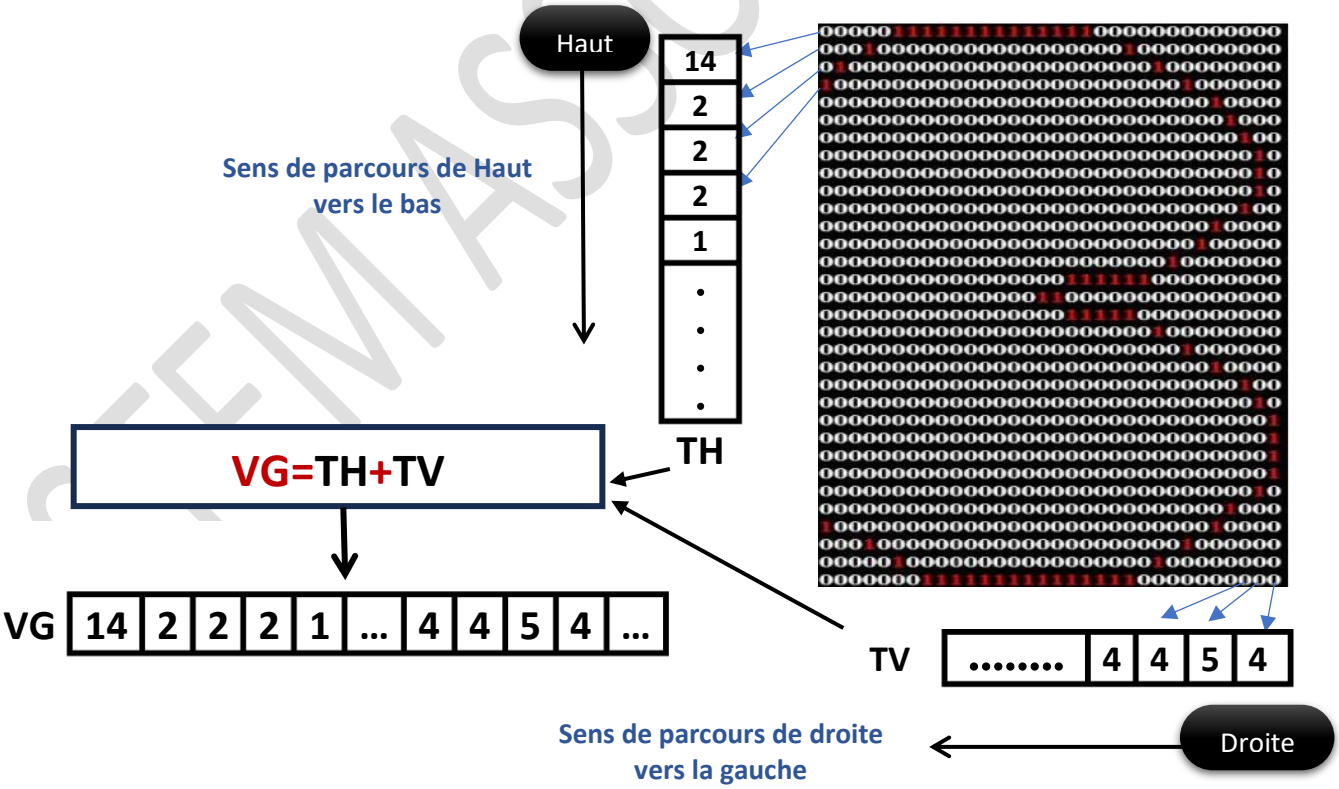
L'étape d'extraction de caractéristiques implique l'utilisation d'une technique d'analyse statistique visant à obtenir des propriétés qui fournissent une description précise de l'image du chiffre manuscrit.

Le processus comprend les étapes suivantes :

1. **Extraction de Caractéristiques : Remplir deux tableaux :**
   - ***Un tableau horizontal (TH) :*** contenant le **nombre** de pixels égal à 1 pour chaque **ligne**. (Avec un sens de parcours de Haut vers le bas).
   - ***Un tableau vertical (TV) :*** contenant le **nombre** de pixels égal à 1 pour chaque **colonne**. (Avec un sens de parcours de droite vers la gauche).
2. **Tableau Global de Caractéristiques :**

Les tableaux de caractéristiques (TH et TV) sont concaténés pour former un vecteur global (VG) représentant l'ensemble des informations extraites.

Exemple :



Tâche4 : **Classification [Détection des Caractéristiques dans une Approche Non Supervisée] :**

Dans cette étape de classification, le processus vise à identifier le chiffre correspondant en explorant le fichier de la base de modèles **"modele.bin",** qui contient les caractéristiques des différents chiffres. Le fichier suit une structure spécifique où chaque enregistrement est composé de deux champs :

- **Chiffre :** Un entier représentant un chiffre entre 0 et 9, indiquant la classe associée aux caractéristiques du chiffre.
- **VE :** Un tableau d'entraînement contenant les caractéristiques descriptives du chiffre, extraites à partir de l'approche non supervisée.

Principe de Classification :

- **Chargement du Fichier de Modèles :** Le programme commence par charger le fichier "modele.bin", ce qui lui permet d'accéder à l'ensemble des caractéristiques et de leurs chiffres associés.
- **Recherche de Correspondance :** Le programme compare le tableau VG avec la liste des tableaux d'entraînement sauvegardés dans le fichier des modèles.
- **Affichage du Résultat :**
  - Si une correspondance est trouvée, le chiffre associé est affiché, fournissant ainsi la prédiction pour le chiffre manuscrit.
  - Si aucune correspondance n'est trouvée dans le fichier des modèles, le programme affiche un message indiquant qu'il est impossible de prédire le chiffre, car il n'existe pas dans la base de modèles.

ANNEXE

Installation et Importation des bibliothèques		
Installer la bibliothèque Pillow	from PIL import Image	**PIL** (Pillow) est utilisé pour manipuler des images

Manipulation des méthodes	
image = Image.open(chemin_image).convert("L")	ouvrir une image depuis le chemin spécifié (**chemin_image**) et la convertir en niveaux de gris (mode "L").
valeur_pixel = image.getpixel((x, y))	la méthode **getpixel()** pour obtenir la valeur du pixel à la position spécifiée **(x, y)**.
C,L = image.size	la propriété **size** pour obtenir les dimensions de l'image.

Mission Bonus

Cette mission bonus vise à exploiter une fonction préalablement développée, permettant de reconnaître un chiffre en fonction d'un nom physique d'une image spécifiée, **afin de déduire un code**.

Ce code est formé à partir des prédictions des chiffres manuscrits associés aux noms des images stockés dans un fichier texte nommé **"code.txt".**

Ce fichier texte, contenu dans le dossier **"ressources",** présente chaque nom physique sur une ligne distincte.

La procédure à suivre pour dévoiler ce code à partir des prédictions est la suivante :

1. Transformer le script solution du **problème2** en une fonction nommée **« Prédire_chiffre(chemin_image) ».**
2. Appliquer la fonction **« Prédire_chiffre(chemin_image) »** pour prédire le code généré par les chiffres manuscrits, les noms physiques étant extraits du fichier "code.txt".
3. Afficher le code résultat de la prédiction.

N.B. : Le candidat n'est pas appelé à remplir le fichier **« code.txt ».**