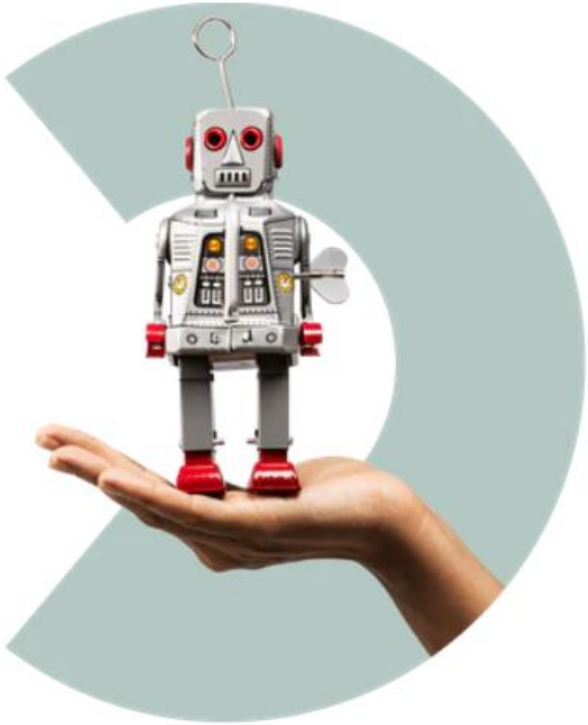
PROJET 1

INTRODUCTION AU METIER D'INGENIEUR IA:
MISE EN OEUVRE DE LA FONCTION POLYGLOTTE D'UN CHATBOT VIA DES SERVICES AZURE

#AZURE COGNITIVE SERVICES #PYTHON



Ingénieur IA

Développez et intégrez des algorithmes de Deep Learning au sein d'un produit IA

OPENCLASSROOMS

OUDDANE NABIL

SOMMAIRE

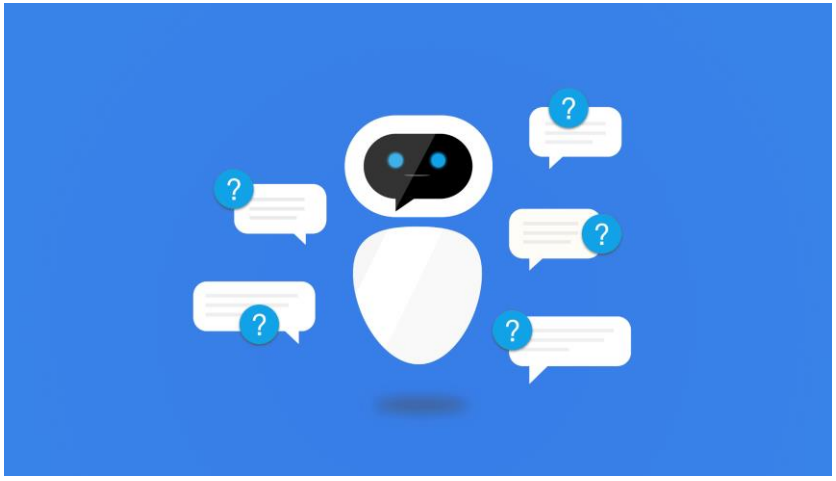
Projet 1

Découvrez le métier d'Ingénieur IA

A INTRODUCTION

1. Contexte

- ENJEU global: Conception d'un chatbot polyglotte, fiable et pertinent pour fluidifier l'experience client du site web de la banque CREDIT



- ENJEU DU P1: Mise en Oeuvre de la fonction polyglotte du chatbot
  - Détection du langage par les modèles pré-entrainés des fonctions cognitives du CLOUD Microsoft AZURE

- SUGGESTION DE RESSOURCES:
  - Jeu de données test Wikipedia Language Identificati Database
  - API CURL de la ressource TRANSLATOR text: https://docs.microsoft.com/en-us/azure/cognitive-services/translator/reference/v3-0-detect

Microsoft Azure

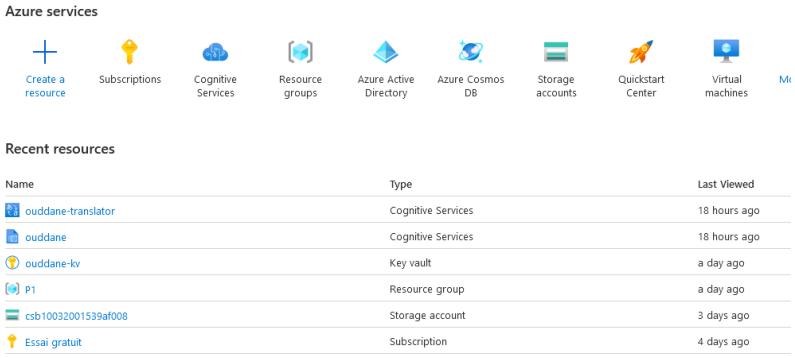
2. Objectifs

- SCRIPT: délivrer du code présentable permettant de
  - Modifier facilement l'input à tester
  - Se connecter de façon sécurisée au service Azure:
    - Les clés ne connexion ne doivent pas être visible
  - Recupérer la prédition du modèle de detection de langue

B PREREQUIS AU PROJET

1. Création d'un compte azure

- Création de la ressource **Microsoft.CognitiveServicesTextAnalytics** dans azure
  - Ce Service possède une fonction de détection de langage bien que non suggérée ainsi qu'une librairie python
- Création de la ressource **Microsoft.CognitiveServicesTranslator** dans azure
  - Ce Service possède une fonction de détection de langage suggérée
  - Son utilisation se fait par une API CURL
- Création de la ressource **Microsoft azure key vault** dans azure
  - Permet de stocker de manière sécurisée les clés d'accés aux API des services : https://docs.microsoft.com/en-us/azure/key-vault/secrets/quick-create-python
  - Utilisation du bash azure CLI en ligne pour la création du coffre fort: https://shell.azure.com/
    - Création du coffre fort en ligne de commande:
      az group create --name KeyVault-PythonQS-rg --location westeurope
      *az keyvault create --name <your-unique-keyvault-name **ouddane-kv**> --resource-group KeyVault-PythonQS-rg*
    - Création d'une application autorisée **P1app** pour le coffre fort dans active directory et parametrage des accés au coffre fort: https://www.c-sharpcorner.com/article/how-to-access-azure-key-vault-secrets-through-rest-api-using-postman/



2. Python

- **INSTALLATION DE LA SUITE ANACONDA**
  - https://www.anaconda.com/
- **CHOIX DE L'ENVIRONMENT JUPYTER NOTEBOOK**
  - Adapté pour les data scientist
- **INSTALLATION DE LIBRAIRIES AZURE VIA JUPYTER:**https://jakevdp.github.io/blog/2017/12/05/installing-python-packages-from-jupyter/
  - azure-ai-textanalytics
    - Service azure cognitive services permettant la detection de langue
  - azure-identity & azure-keyvault-secrets
    - Service permettant de gérer l'accés au coffre fort azure et la connexion sécurisée aux services azure

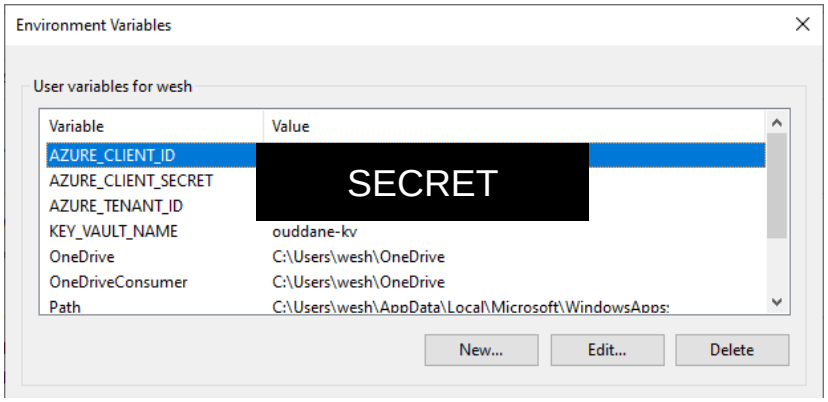
STEP 0.1: INSTALLATION DES LIBRAIRIES AZURE NECESSAIRES VIA JUPYTER

Installation de **azure-ai-textanalytics** pour l'analyse de texte IA Installation de **azure-identity** et **azure-keyvault-secrets** pour les accés au coffre fort AZURE https://jakevdp.github.io/blog/2017/12/05/installing-python-packages-from-jupyter/

```
#aminata1902@hotmail.fr
#STEP 0.1
#prerequisite (one time)

import sys
!{sys.executable} -m pip install azure-ai-textanalytics

#creation de la ressource Microsoft.CognitiveServicesTextAnalytics dans azure
#https://docs.microsoft.com/en-us/azure/cognitive-services/text-analytics/quickstarts/client-libraries-rest-api?tabs=ver
#https://azure.microsoft.com/fr-fr/blog/getting-started-with-cognitive-services-language-understanding-container/

!{sys.executable} -m pip install azure-identity
!{sys.executable} -m pip install azure-keyvault-secrets
```

3. Création de variables d'environnement permettant l'accés au coffre fort via l'application P1APP autorisée

- **Création des clés suivantes disponibles sur son compte azure paramétré**
  - AZURE_CLIENT_ID
  - AZURE_CLIENT_SECRET
  - AZURE_TENANT_ID
  - KEY_VAULT_NAME
- **Insertion ou modification de ses valeurs par un code python si besoin**



- la fonction **os.environ** map les variables environmentales une fois pour toutes à l'import de la librairie os. Les changements de l'environnement opérés après ce mapping ne sont pas répercutés dans os.environ, à part les modifications directes de os.environ dans le code.

```python
#STEP 0.3
#environ var set up

import os

#something weird
#need first to set up the environment variables on the system (control panel)


os.environ['KEY_VAULT_NAME'] = 'ouddane-kv'
os.environ['AZURE_TENANT_ID']=
#https://docs.microsoft.com/fr                                  active-directory-how-to-find-tenant
os.environ['AZURE_CLIENT_ID']=
#https://www.c-sharpcorner.com                                  ecrets-through-rest-api-using-postman/
os.environ['AZURE_CLIENT_SECRET

#import os
#print(os.getenv('KEY_VAULT_NAME'))
#print(os.getenv('AZURE_TENANT_ID'))
#print(os.getenv('AZURE_CLIENT_ID'))
#print(os.getenv('AZURE_CLIENT_SECRET'))
#print(os.environ)
```

4. Insertion des clés API des services dans le coffre fort AZURE KEYVAULT

- Création d'un script python permettant de demander les clés puis de les stocker
  - L'accés est complètement sécurisé
  - **STEP 0.4: STOCKAGE DE LA KEY cognitives services DANS LE COFFRE FORT AZURE KEYVAULT**

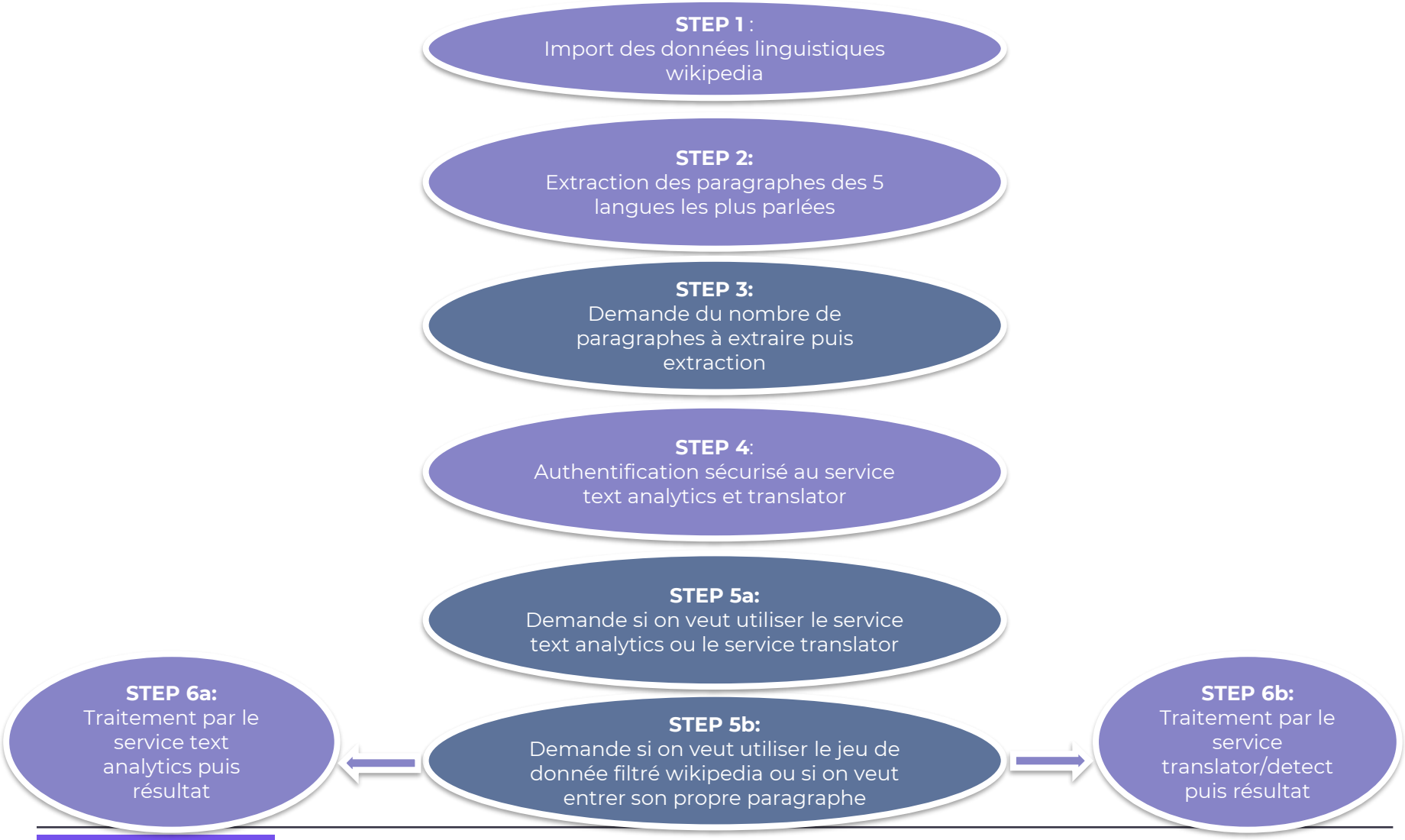
    https://docs.microsoft.com/fr-fr/azure/key-vault/secrets/quick-create-python
    https://www.c-sharpcorner.com/article/how-to-access-azure-key-vault-secrets-through-rest-api-using-postman/

```
:
#STEP 0.4
#code to store keys within azure keyvaults
import os
import cmd
from azure.keyvault.secrets import SecretClient
from azure.identity import DefaultAzureCredential

keyVaultName = os.getenv('KEY_VAULT_NAME')
KVUri = f"https://{keyVaultName}.vault.azure.net"

#https://docs.microsoft.com/en-us/python/api/overview/azure/identity-readme?view=azure-python

credential = DefaultAzureCredential()
client = SecretClient(vault_url=KVUri, credential=credential)

secretName = input("Input a name for your secret > ")
secretValue = input("Input a value for your secret > ")

print(f"Creating a secret in {keyVaultName} called '{secretName}' with the value '{secretValue}' ...")

client.set_secret(secretName, secretValue)

print(" done.")
```

```
Input a name for your secret > test
Input a value for your secret > test
Creating a secret in ouddane-kv called 'test' with the value 'test' ...
 done.
```

SCRIPT

1. Fonctionnement du script
Possibilité d'utiliser 2 services azure différents



2. Import des données linguistiques wikipedia

- Utilisation des fonctions de la librairie pandas
  - ne pas oublier la petite option quoting=3 dans read.table() qui fait la différence pour télécharger tous les lignes proprement

```
#step 1
#wikipedia data languages reading
#test avec les dataframes de pandas

#When reading the txt file with pandas' read_csv, some line loss occurred.
#The reason is that there is a single English quotation mark in a line, causing the \n line break to be invalid.
#Multiple lines are concatenated until the next single quote is encountered
#add option quoting=3 to tackle the issue :QUOTE_NONE

import pandas as pd
import numpy as np

dflabel = pd.read_csv('input/labels.csv', delimiter = ';')
dfx1 = pd.read_table('input/x_train.txt', delimiter = None, header=None, encoding='utf-8', quoting=3)
dfy1 = pd.read_table('input/y_train.txt', delimiter = None, header=None, encoding='utf-8', quoting=3)
dfx2 = pd.read_table('input/x_test.txt', delimiter = None, header=None, encoding='utf-8', quoting=3)
dfy2 = pd.read_table('input/y_test.txt', delimiter = None, header=None, encoding='utf-8', quoting=3)

#row number quick checking
print(len(dfx1.index))
print(len(dfx2.index))
print(len(dfy1.index))
print(len(dfy2.index))
print(len(dflabel.index))
```

```
117500
117500
117500
117500
235
```

3. Extraction des paragraphes des 5 langues les plus parlées

- Utilisation des fonctions de la librairie pandas
  - Selon WIKIPEDIA les 5 langues les plus parlées sont les suivantes: english / mandarin / hindi / spanish / standard arabic: https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers

```
:
#step 2
#5 most spoken languages selection
#https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers
#english / mandarin / hindi / spanish / standard arabic

#sentences and language merging
df1 = pd.concat([dfx1, dfy1], axis=1)
df1.columns=["para","langcod"]
df2 = pd.concat([dfy2, dfy2], axis=1)
df2.columns=["para","langcod"]

#5 languages extraction from labels
dflabel5mml=dflabel.loc[(dflabel['English']=='Arabic')|
                        (dflabel['English']=='English')|
                        (dflabel['English']=='Hindi')|
                        (dflabel['English']=='Spanish')|
                        (dflabel['English']=='Standard Chinese')]
dflabel5mml2=dflabel5mml.rename(columns={'Label':'langcod'})
dflabel5mml3=dflabel5mml2[['langcod','English']]

#sentences selection by most spoken languages-> merging
df1s = pd.merge(df1, dflabel5mml3, how="inner", on=["langcod"])
df2s = pd.merge(df2, dflabel5mml3, how="inner", on=["langcod"])

#row number quick checking
print(len(df1s.index))
print(len(df2s.index))
```

```
2500
2500
```

4. EXTRACTION DE N PARAGRAPHES ALEATOIRES POUR NE PAS EXPLOSER LES COMPTEURS AZURE

- Demande du nombre de paragraphes à extraire des données sources préfiltrées
  - Nombre minimum de 5 pour obtenir les 5 langues.
  - Tirage aléatoire de N paragraphes provenant des 2500 paragraphes prefiltrés.
    - Tirage équitable dans les 5 langues
  - Récupération également des langue correspondantes d'après wikipedia

```
#step 3
#extraction of n random sentences form dataframe
import random

def rand_shrink_df(df):

    size = input("Input a number of paragraphs, bigger than 5, that you want to test ,\
keep in mind that azure is not free > ")
    sizeint = int(size)
    #Generate random numbers
    #randomlist = random.sample(range(0, len(df.index)-1), sizeint)
    randomlist = random.sample(range(0, 500), sizeint)
    #print(randomlist)
    #trick to get all the language
    factor = [(i % 5)*500 for i in range(sizeint)]
    #print(factor)
    #operation of 2 list with zip
    product = [x+y for x,y in zip(randomlist,factor)]
    #print(product)



    for i in range(sizeint):
                            #attention pour que le dataframe reste en row, penser aux double crochets
        dfa=df.iloc[[product[i]],:]
        #print(i)
        if i==0 :
            dfaa=dfa
            #print(dfaa)
            #print(len(dfaa.index))
        else:
            dfaa=dfaa.append(dfa)
            #print(len(dfa.index))
    return dfaa

df2az=rand_shrink_df(df1s)
#print(len(df2az.index))
print(df2az)

#faire en sorte que toutes les langues apparaissent dans le tirage
```

```
Input a number of paragraphs, bigger than 5, that you want to test ,keep in mind that azure is not free >
```

5. A) Authentification sécurisée au service text analytics

- Connexion sécurisé au service azure keyvault
  - Récupération sécurisé de la Key API permettant la connexion au service text analytics
  - Aucune clé ne figure dans le code, des variables d'environment sont utilisées
- Connexion sécurisée au service text analytics grâce a la key API

```python
#step 4a
#authenticate the client
#azure key vault (coffre fort)  to securely store keys
from azure.keyvault.secrets import SecretClient
from azure.identity import DefaultAzureCredential

keyVaultName = os.getenv('KEY_VAULT_NAME')
KVUri = f"https://{keyVaultName}.vault.azure.net"

#https://docs.microsoft.com/en-us/python/api/overview/azure/identity-readme?view=azure-python

credential = DefaultAzureCredential()
#get the needed environmental variables to connect to the keyvault
client = SecretClient(vault_url=KVUri, credential=credential)

secretName = "keytextanalytics"
endpoint = "https://ouddane.cognitiveservices.azure.com/"

#https://docs.microsoft.com/en-us/azure/cognitive-services/text-analytics/quickstarts/client-libraries-rest-api?tabs=ver
# use this code if you're using SDK version is 5.0.0
from azure.ai.textanalytics import TextAnalyticsClient
from azure.core.credentials import AzureKeyCredential

def authenticate_client():
    key=client.get_secret(secretName)
    ta_credential = AzureKeyCredential(key.value)
    text_analytics_client = TextAnalyticsClient(
            endpoint=endpoint,
            credential=ta_credential)
    return text_analytics_client

client = authenticate_client()
```

5. B) Authentification sécurisée au service translator

- Connexion sécurisé au service azure keyvault
  - Récupération sécurisé de la Key API permettant la connexion au service translator
  - Aucune clé ne figure dans le code, des variables d' environment sont utilisées
- La connexion sécurisée au service translator se fait en envoyant la requête POST de détection de langage en passant la clé API comme paramètre headers

```python
#step 6
#language detection by translator
import os
import cmd
import requests, uuid, json

#authenticate the client
#azure key vault (coffre fort)  to securely store keys
from azure.keyvault.secrets import SecretClient
from azure.identity import DefaultAzureCredential

def azure_translator_language_detection(df):


    #https://docs.microsoft.com/en-us/python/api/overview/azure/identity-readme?view=azure-python
    credential = DefaultAzureCredential()
    #get the needed environmental variables to connect to the keyvault
    keyVaultName = os.getenv('KEY_VAULT_NAME')
    KVUri = f"https://{keyVaultName}.vault.azure.net"
    client = SecretClient(vault_url=KVUri, credential=credential)

    #get the key frome the azure key vault to connect to translator
    secretName = "keytranslator"
    key=client.get_secret(secretName)

    #https://docs.microsoft.com/en-us/azure/cognitive-services/translator/quickstart-translator?tab

    # Add your subscription key and endpoint
    subscription_key = key.value
    endpoint = "https://api.cognitive.microsofttranslator.com/detect?api-version=3.0"
    location = "francecentral"

    headers = {
        'Ocp-Apim-Subscription-Key': subscription_key,
        'Content-type': 'application/json',
        'Ocp-Apim-Subscription-Region': location,
        'X-ClientTraceId': str(uuid.uuid4())
    }
```

6. A) Détection du langage via l'IA TEXT ANALYTICS d'AZURE

- Demande du type d'input à tester
  - Paragraphe 'on the fly' : On copie colle un paragraphe et on récupère la langue après requête
  - table préfiltrée des paragraphes wikipidia : On récupère un pourcentage moyen, très bon, de matching avec les valeurs de langue wikipédia et on affiche les paragraphes posant problème .

```python
#step 5
#language detection by text analytics
def azure_text_analytics_language_detection(client,df):

    own= input("Do you want to test your own sentence? write Y or N > ")
    if own=="Y":
        doc= [input("paste your sentence > ")]
        response = client.detect_language(documents = doc, country_hint = "")[0]
        print("According to AZURE text analytics, the language of your sentence is ",response.primary_language.name )
    else:
        df["azure"] = ""
        df["flag"]= 0
        for i in range(len(df.index)):
            try:
                documents = [df.iloc[i,0]]
                response = client.detect_language(documents = documents, country_hint = "")[0]
                df.iloc[i,3]=response.primary_language.name
                if (df.iloc[i,3]== "Chinese_Traditional") | (df.iloc[i,3]== "Chinese_Simplified"):
                    df.iloc[i,3]= "Standard Chinese"
                if df.iloc[i,3]== df.iloc[i,2]:
                    df.iloc[i,4]=1
                else:
                    df.iloc[i,4]=0
            except Exception as err:
                print("Encountered exception. {}".format(err))

        print("La detection fonctionne à",df["flag"].mean()*100,"%")
        if df["flag"].mean()<1:
            print("L'erreur provient peut etre de la donnée source")
            df0=df.loc[(df['flag']==0)]
            print(df0)

azure_text_analytics_language_detection(client,df2az)
```

```
Do you want to test your own sentence? write Y or N > Y
paste your sentence > Au menu de la journée, deux huitièmes de finale : le premier entre les Pays-Bas et la République tché
que, le second entre la Belgique et le Portugal. Chaque matin, retrouvez l'actualité de la seizième édition de l'Euro de fo
otball.
According to AZURE, the language of your sentence is  French
```

6. B) Détection du langage via l'IA TRANSLATOR/DETECT

- Demande du type d'input à tester
  - Paragraphe 'on the fly' : On copie colle un paragraphe et on récupère la langue après requête. On peut facilement afficher le pourcentage de confiance de la détection
  - table préfiltrée des paragraphes wikipidia : On récupère un pourcentage moyen, très bon, de matching avec les valeurs de langue wikipédia , un pourcentage moyen de confiance de la détéction fourni par azure et on affiche les paragraphes posant problème .

```
    own= input("Do you want to test your own sentence? write Y or N > ")
    if own=="Y":
        doc= input("paste your sentence > ")
        # You can pass more than one object in json body.
        body = [{
            'text': doc
        }]
        ## lib requests
        ## post sends a POST request (http) to the specified url and is used when you want to send some dat
        request = requests.post(endpoint, headers=headers, json=body)
        ##response.json() returns a JSON object of the result / Whenever we make a request to a specified U
        response = request.json()

        #json.dumps() function converts a Python object into a json string
        response2 = json.dumps(response)
        #json.loads() method can be used to parse a valid JSON string and convert it into a Python Dictiona
        dictresponse = json.loads(response2)[0]

        #print(dictresponse)
        print("According to AZURE text analytics, the language of your sentence is ", dictresponse['languag

    else:
        #init flag
        df["azure"] = ""
        df["flag"]= 0
        df["azscore"]=0
        for i in range(len(df.index)):
           # try:
            doc = df.iloc[i,0]
            body = [{
                    'text': doc
             }]
            request = requests.post(endpoint, headers=headers, json=body)
            response = request.json()
            response2 = json.dumps(response)
            dictresponse = json.loads(response2)[0]
            #print(dictresponse)
            df.iloc[i,3] = dictresponse['language']
            df.iloc[i,5] = dictresponse['score']
            if (((df.iloc[i,3]== "en") & (df.iloc[i,2]== "English"))|
                ((df.iloc[i,3]== "ar") & (df.iloc[i,2]== "Arabic"))|
                ((df.iloc[i,3]== "es") & (df.iloc[i,2]== "Spanish"))|
                ((df.iloc[i,3]== "hi") & (df.iloc[i,2]== "Hindi"))|
                ((df.iloc[i,3]== "zh-Hant") & (df.iloc[i,2]== "Standard Chinese"))|
                ((df.iloc[i,3]== "zh-Hans") & (df.iloc[i,2]== "Standard Chinese"))):
                df.iloc[i,4]=1
            else:
                df.iloc[i,4]=0
           # except Exception as err:
           #     print("Encountered exception. {}".format(err))

        print("La detection fonctionne à",df["flag"].mean()*100,"% selon notre indicateur et le score moyen
        if df["flag"].mean()<1:
            print("L'erreur provient peut etre de la donnée source: mauvais flag ou plusieurs langues dsur
            #65417 de train.txt: plusieurs langues
            df0=df.loc[(df['flag']==0)]
            print(df0)

azure_translator_language_detection(df2az)
```

D CONCLUSION

1. Les Conclusions

- Fiabilité des services de détection de langue d'azure
  - Aussi bien pour le service text analytics que pour le service translator
- Les données sources wikipidia ne sont pas 100% propres
  - Paragraphes avec plusieurs langues
  - Langue associée fausse : exemple: ligne 11251 de train.txt reference hindi alors que c est de l'anglais
- Projet
  - Formateur sur python , azure et d'autres librairies
  - La réalisation répond à la demande et va au delà en testant 2 services différents