

Partie1: Tutorial

1. Introduction:

Cette partie présente un tutoriel et une démonstration pratique du processus ETL (Extraction, Transformation, Chargement) appliqué aux documents. L'objectif est de convertir des données non structurées en informations exploitables et visualisables.

- **Tutoriel** : Explication des étapes du processus ETL à l'aide des scripts, couvrant l'ingestion, le traitement et le stockage des données.
- **Démonstration** : Mise en œuvre du pipeline ETL dans un scénario réel à travers trois applications interconnectées de manière flexible, automatisant ainsi le processus depuis l'ingestion des données jusqu'à leur visualisation.

Cette partie associe théorie et pratique pour illustrer de manière claire et efficace les étapes du processus ETL.

2. Les outils utilisé:

- **Telegram API** : Pour l'ingestion de données provenant de messages et documents via Telegram.
- **EasyOCR** : Outil OCR utilisé pour extraire du texte à partir d'images et de documents scannés.
- **Tesseract (pytesseract)** : Autre outil OCR pour l'extraction de texte à partir d'images.
- **spaCy** : Modèle NLP (traitement du langage naturel) pour analyser et structurer les données textuelles.
- **PyPDF2** : Pour extraire du texte à partir de fichiers PDF.
- **pdfplumber** : Une alternative à PyPDF2 pour une extraction précise de texte depuis les PDFs.
- **regex (expressions régulières)** : Pour extraire, nettoyer et structurer des données textuelles avec des motifs précis.
- **Flask** : Framework web léger pour développer une application interactive d'ETL et d'automatisation.
- **Python** : Langage principal pour le développement des scripts ETL.
- **Data Warehouse** : Stockage des données structurées pour un accès rapide et une gestion centralisée.
- **Dashboard de Visualisation** : Présentation des données analysées sous forme de graphiques pour une prise de décision.

3. Prétraitement d'Image pour l'Optimisation de l'OCR :

L'objectif est l'amélioration de la qualité des images (niveaux de gris, contraste, redressement, suppression du bruit) pour des résultats OCR plus précis.

1.3 Conversion en niveaux de gris :

```
import cv2
import matplotlib.pyplot as plt

# Charger l'image
image = cv2.imread(r"C:\Users\dell\Desktop\BI_prototype\tutorial\text_with_c.jpg")
# Convertir l'image en niveaux de gris
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

plt.figure(figsize=(10, 5))
# Afficher l'image originale
plt.subplot(1, 2, 1) # (nrows, ncols, index)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB)) # Convertir de BGR à RGB pour affichage
plt.title("Image Originale")
plt.axis('off')

# Afficher l'image en niveaux de gris
plt.subplot(1, 2, 2)
plt.imshow(gray_image, cmap='gray') # Affichage en niveaux de gris
plt.title("Image en Niveaux de Gris")
plt.axis('off')
# Afficher les deux images côte à côte
plt.show()
```

Image Originale



Image en Niveaux de Gris



2.3 Binarisation (Seuillage) :

```
binary_image = cv2.threshold(gray_image, 220, 255, cv2.THRESH_BINARY)[1]
plt.figure(figsize=(10, 5))
# Afficher l'image originale
plt.subplot(1, 2, 1) # (nrows, ncols, index)
plt.imshow(cv2.cvtColor(gray_image, cv2.COLOR_BGR2RGB)) # Convertir de BGR à RGB pour affichage
plt.title("Image Originale")
plt.axis('off')

# Afficher l'image en niveaux de gris
plt.subplot(1, 2, 2)
plt.imshow(binary_image, cmap='gray') # Affichage en niveaux de gris
plt.title("Image en Niveaux de Gris")
plt.axis('off')
# Afficher les deux images côte à côte
plt.show()
```

Image Originale



Image binarisé



3.3 Élimination du bruit :

Note :cette code ne fait aucun réduction de bruit , pour un simple raison, c'est le fait de réduire le bruit nécessite un model AI bien entrainé sur des tache de (denoising), soit d'utilisé une distributions des pixels qui suit la loi normale mais cette méthode très pauvre pour des cas similaire comme la suite :

```
## denoised image est une resultat d'un model AI pre-entraine ,pas d'un traitement dans cette cellule de code

denoised= cv2.imread(r"C:\Users\dell\Desktop\BI_prototype\tutorial\Fontfre_Clean_TE.png")
noised=cv2.imread(r"C:\Users\dell\Desktop\BI_prototype\tutorial\Fontfre_Noise_RE.png")

plt.figure(figsize=(10, 5))
# Afficher l'image originale
plt.subplot(1, 2, 1) # (nrows, ncols, index)
plt.imshow(cv2.cvtColor(noised, cv2.COLOR_BGR2RGB)) # Convertir de BGR à RGB pour affichage
plt.title("Image Originale")
plt.axis('off')

# Afficher l'image en niveaux de gris
plt.subplot(1, 2, 2)
plt.imshow(denoised, cmap='gray') # Affichage en niveaux de gris
plt.title("Image en Niveaux de Gris")
plt.axis('off')
# Afficher les deux images côte à côte
plt.show()
```

Image Originale

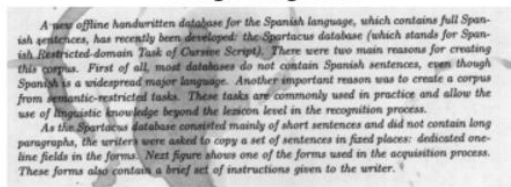


Image sans bruit

A new offline handwritten database for the Spanish language, which contains full Spanish sentences, has recently been developed: the Spartacus database (which stands for Spanish Restricted-domain Task of Cursive Script). There were two main reasons for creating this corpus. First of all, most databases do not contain Spanish sentences, even though Spanish is a widespread major language. Another important reason was to create a corpus from semantic-restricted tasks. These tasks are commonly used in practice and allow the use of linguistic knowledge beyond the lexicon level in the recognition process. As the Spartacus database consisted mainly of short sentences and did not contain long paragraphs, the writers were asked to copy a set of sentences in fixed places: dedicated one-line fields in the forms. Next figure shows one of the forms used in the acquisition process. These forms also contain a brief set of instructions given to the writer.

4.3 Redimensionnement :

```
width = 800 # Largeur cible
height = 600 # Hauteur cible

resized_image = cv2.resize(denoised, (width, height), interpolation=cv2.INTER_CUBIC)
resized_image = cv2.threshold(resized_image, 220, 255, cv2.THRESH_BINARY)[1]
plt.figure(figsize=(10, 5))
# Afficher l'image originale
plt.subplot(1, 2, 1) # (nrows, ncols, index)
plt.imshow(cv2.cvtColor(denoised, cv2.COLOR_BGR2RGB)) # Convertir de BGR à RGB pour affichage
plt.title("Image Originale")
plt.axis('off')

# Afficher l'image en niveaux de gris
plt.subplot(1, 2, 2)
plt.imshow(resized_image, cmap='gray') # Affichage en niveaux de gris
plt.title("Image en Niveaux de Gris")
plt.axis('off')
# Afficher les deux images côte à côte
plt.show()
```

Image Originale

A new offline handwritten database for the Spanish language (Spanish Restricted-domain Task of Cursive Script). There were two this corpus. First of all, most databases do not contain Spanish. Spanish is a widespread major language. Another important reason from semantic-restricted tasks. These tasks are commonly used use of linguistic knowledge beyond the lexicon level in the recognition

As the Spartacus database consisted mainly of short sentence paragraphs, the writers were asked to copy a set of sentences in fine line fields in the forms. Next figure shows one of the forms used. These forms also contain a brief set of instructions given to the

Image apres Redimensionnement

A new offline handwritten database for the Spanish language (Spanish Restricted-domain Task of Cursive Script). There were two this corpus. First of all, most databases do not contain Spanish. Spanish is a widespread major language. Another important reason from semantic-restricted tasks. These tasks are commonly used use of linguistic knowledge beyond the lexicon level in the recognition

As the Spartacus database consisted mainly of short sentence paragraphs, the writers were asked to copy a set of sentences in fine line fields in the forms. Next figure shows one of the forms used. These forms also contain a brief set of instructions given to the

5.3 Correction d'inclinaison (Deskewing) :

```
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Charger l'image
image_path = r"C:\Users\dell\Desktop\BI_prototype\tutorial\text_with_c - Copy.jpg"
binary_image = cv2.imread(image_path)

# Convertir en niveaux de gris
gray_image = cv2.cvtColor(binary_image, cv2.COLOR_BGR2GRAY)

# Appliquer un seuillage binaire (nécessaire pour `findNonZero`)
_, binary_thresh = cv2.threshold(gray_image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Trouver les coordonnées des pixels non nuls
coords = cv2.findNonZero(binary_thresh)

# Calculer l'angle de rotation
angle = cv2.minAreaRect(coords)[-1]
if angle < -45:
    angle = (90 + angle)
else:
    angle = -angle

# Calculer la matrice de rotation
(h, w) = binary_image.shape[:2] # Dimensions de l'image
rotation_matrix = cv2.getRotationMatrix2D((w / 2, h / 2), angle, 1)

# Effectuer la rotation pour redresser l'image
deskewed_image = cv2.warpAffine(binary_image, rotation_matrix, (w, h))

# Afficher les résultats
plt.figure(figsize=(10, 5))

# Afficher l'image originale
plt.subplot(1, 2, 1) # (nrows, ncols, index)
```

Image Originale

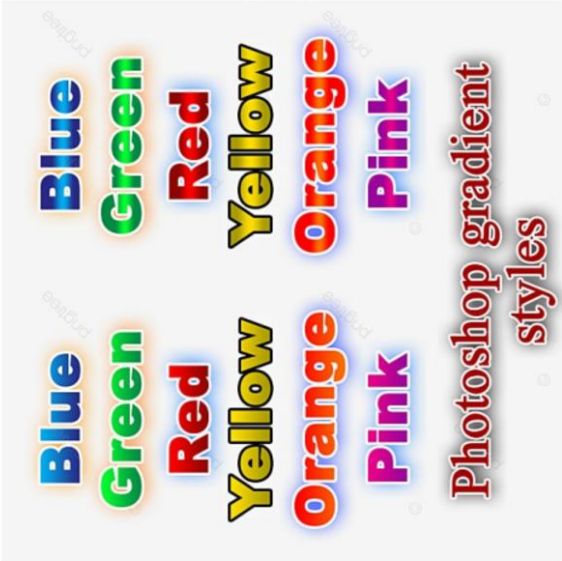


Image Redressée



6.3 Amélioration du contraste:

```
image=r"C:\Users\dell\Desktop\BI_prototype\tutorial\text_with_c.jpg"
image=cv2.imread(image)
image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
enhanced_image = cv2.equalizeHist(image)
plt.figure(figsize=(10, 5))
# Afficher l'image originale
plt.subplot(1, 2, 1) # (nrows, ncols, index)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB)) # Convertir de BGR à RGB pour affichage
plt.title("Image Originale")
plt.axis('off')

# Afficher l'image en niveaux de gris
plt.subplot(1, 2, 2)
plt.imshow(enhanced_image, cmap='gray') # Affichage en niveaux de gris
plt.title("Image apres Amélioration du contraste")
plt.axis('off')
# Afficher les deux images côte à côte
plt.show()
```

Image Originale



Image apres Amélioration du contraste



4. Data extraction :

Pytesseract : Utilisation de Tesseract OCR via la bibliothèque pytesseract pour extraire du texte à partir d'images, efficace pour les documents avec texte imprimé clair.

EasyOCR : Utilisation de la bibliothèque EasyOCR pour l'extraction de texte à partir d'images, idéale pour des documents avec des polices non standards ou de mauvaise qualité.

Extraction de texte à partir de PDF : Utilisation de bibliothèques comme PyPDF2 ou pdfplumber pour extraire le texte des fichiers PDF, notamment pour les factures ou rapports stockés sous ce format.

1.4 Pytesseract :

- **Installation** : Installez Pytesseract via `pip install pytesseract` et configurez le chemin vers l'exécutable Tesseract (à installer aussi) avec

`pytesseract.pytesseract.tesseract_cmd`, ou ajouter le a une path avec les variables d'environnement

- **Extraction de texte** : Utilisez `pytesseract.image_to_string(image)` pour extraire le texte d'une image après l'avoir chargée avec PIL.
- **Support des langues** : Ajoutez des langues comme le français avec `lang='fra'` pour une meilleure précision de la reconnaissance.
- **Options avancées** : Personnalisez les paramètres de Tesseract avec des options comme `--psm` et `--oem` pour ajuster la segmentation et le moteur OCR.


Exemple :

```
#1. Pytesseract (pour extraire du texte à partir d'une image)
import pytesseract
from PIL import Image
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
# Charger L'image
image_path=r"C:\Users\dell\Desktop\BI_prototype\tutorial\purchasse.png"
image = Image.open(image_path)
# Extraire Le texte à partir de L'image
text1 = pytesseract.image_to_string(image)

print("Texte extrait avec Pytesseract :")
print(text1)
```

Résultat :(le code précédente n'inclut pas la visualisation qui figure ici)

Image originale



Purchase Orders

Order ID	Order Date	Customer Name
10250	2016-07-08	Mario Pontes

Products

Product ID:	Product:	Quantity:	Unit Price:
41	Jack's New England Clam Chowder	10	7.7
51	Manjimup Dried Apples	35	42.4
65	Louisiana Fiery Hot Pepper Sauce	15	16.8

Texte extrait

@

Purchase Orders

Order ID Order Date Customer Name
10250 2016-07-08 Mario Pontes

Products

Product ID: Product: Quantity Unit Price:
41 Jack's New England Clam Chowder 10 7.7
51 Manjimup Dried Apples 35 42.4
65 Louisiana Fiery Hot Pepper Sauce 15 16.8

2.4 EasyOCR :

- **Installation** : Installez EasyOCR via `pip install easyocr` pour commencer à utiliser la bibliothèque.

- **Initialisation du lecteur OCR** : Créez un lecteur OCR avec `easyocr.Reader(['en', 'fr'])` en spécifiant les langues pour l'extraction de texte.
- **Extraction de texte** : Utilisez `reader.readtext(image_path)` pour extraire le texte d'une image, où `image_path` est le chemin vers l'image.
- **Résultats personnalisés** : Les résultats sont renvoyés sous forme de liste, incluant les coordonnées de la zone et le texte détecté, permettant de traiter les informations de manière flexible.

Exemple :

```
#2. EasyOCR (pour extraire du texte à partir d'une image)
import easyocr
# Initialiser l'outil OCR
reader = easyocr.Reader(['en']) # 'en' pour l'anglais

# Extraire le texte à partir de l'image
results = reader.readtext(image_path)

print("Texte extrait avec EasyOCR :")
text2="" ""
for result in results:
    print(result[1]) # Le texte extrait est dans le deuxième élément de la tuple
    text2+=result[1]
```

Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU.

Texte extrait avec EasyOCR :


Purchase Orders

Order ID

Order Date

Order Date

Résultat :

Image originale		Texte extrait	
			
Purchase Orders		Purchase Orders	
Order ID	Order Date	Order ID	Order Date
10250	2016-07-08	Customer Name	10250
		2016-07-08	Mario Pontes
		Products	Product ID:
		Product:	Quantity:
		Unit Price:	
		41	Jack's New England Clam Chowder
		10	7.7
		51	Manjimup Dried Apples
		35	42.4
		65	Louisiana
		Hot Pepper Sauce	15
		16.8	Fiery

3.4 PyPDF2

- **Installation** : Installez PyPDF2 via `pip install PyPDF2` pour pouvoir manipuler des fichiers PDF.

- **Ouverture du fichier PDF** : Utilisez `PyPDF2.PdfReader(file)` pour ouvrir un fichier PDF en mode lecture.
- **Extraction du texte** : Récupérez le texte d'une page avec `page.extract_text()`, où page est une page spécifique du document PDF.
- **Manipulation des pages** : Vous pouvez parcourir toutes les pages du PDF et extraire le texte de chacune pour obtenir un document complet.

Exemple :

```
#3. Extraction de texte à partir de PDF (en utilisant PyPDF2)
import PyPDF2
# Ouvrir le fichier PDF
pdf_path=r"C:\Users\dell\Desktop\BI_prototype\tutorial\invoice_10248.pdf"
with open(pdf_path, 'rb') as file:
    reader = PyPDF2.PdfReader(file)

    # Extraire le texte de chaque page
    text2 = ""
    for page in reader.pages:
        text2 += page.extract_text()

print("Texte extrait du PDF avec PyPDF2 :")
print(text2)
```

Texte extrait du PDF avec PyPDF2 :
 Invoice
 Order ID: 10248
 Customer ID: VINET
 Order Date: 2016-07-04

Résultat :

Première page du PDF

Invoice

Order ID: 10248

Customer ID: VINET

Order Date: 2016-07-04

Customer Details:

Contact Name:	Paul Henriot
Address:	59 rue de l'Abbaye
City:	Reims
Postal Code:	51100
Country:	France
Phone:	26.47.15.10
Fax:	26.47.15.11

Product Details:

Product ID	Product Name	Quantity	Unit Price
11	Queso Cabrales	12	14.0
42	Singaporean Hokkien Fried Mee	10	9.8
72	Mozzarella di Giovanni	5	34.8
		TotalPrice	440.0

Texte extrait du PDF

Invoice
 Order ID: 10248
 Customer ID: VINET
 Order Date: 2016-07-04
 Customer Details:
 Contact Name: Paul Henriot
 Address: 59 rue de l'Abbaye
 City: Reims
 Postal Code: 51100
 Country: France
 Phone: 26.47.15.10
 Fax: 26.47.15.11
 Product Details:
 Product ID Product Name Quantity Unit Price
 11 Queso Cabrales 12 14.0
 42 Singaporean Hokkien Fried Mee 10 9.8
 72 Mozzarella di Giovanni 5 34.8
 TotalPrice 440.0
 Page 1

4.4 Pdfplumber :

- **Installation** : Installez pdfplumber avec la commande `pip install pdfplumber` pour accéder à ses fonctionnalités d'extraction de texte.
- **Ouverture du fichier PDF** : Utilisez `pdfplumber.open(file_path)` pour ouvrir un fichier PDF et accéder à ses pages.
- **Extraction du texte** : Utilisez `page.extract_text()` pour extraire le texte d'une page spécifique dans le PDF.
- **Accès aux éléments de la page** : pdfplumber permet également d'extraire des informations sur les tables, les images et les autres éléments visuels présents dans le PDF.

Exemple :

```
|: #3. Extraction de texte à partir de PDF (en utilisant pdfplumber)
import pdfplumber
# Ouvrir le fichier PDF
pdf2_path=r"C:\Users\dell\Desktop\BI_prototype\tutorial\invoice_10257.pdf"
with pdfplumber.open(pdf2_path) as pdf:
    # Extraire le texte de la première page
    page = pdf.pages[0]
    text3 = page.extract_text()

print("Texte extrait du PDF avec pdfplumber :")
print(text3)
```

Texte extrait du PDF avec pdfplumber :
Invoice
Order ID: 10257
Customer ID: HILAA
Order Date: 2016-07-16
Customer Details:

Résultat :

Première page du PDF

Invoice

Order ID: 10257

Customer ID: HILAA

Order Date: 2016-07-16

Customer Details:

Contact Name:	Carlos Hernández
Address:	Carrera 22 con Ave. Carlos Soublette #8-35
City:	San Cristóbal
Postal Code:	5022
Country:	Venezuela
Phone:	(5) 555-1340
Fax:	(5) 555-1948

Product Details:

Product ID	Product Name	Quantity	Unit Price
27	Schoggi Schokolade	25	35.1
39	Chartreuse verte	6	14.4
77	Original Frankfurter grüne Soße	15	10.4
TotalPrice			1119.9

Texte extrait du PDF

Invoice
Order ID: 10257
Customer ID: HILAA
Order Date: 2016-07-16
Customer Details:
Contact Name: Carlos Hernández
Address: Carrera 22 con Ave. Carlos Soublette #8-35
City: San Cristóbal
Postal Code: 5022
Country: Venezuela
Phone: (5) 555-1340
Fax: (5) 555-1948
Product Details:
Product ID Product Name Quantity Unit Price
27 Schoggi Schokolade 25 35.1
39 Chartreuse verte 6 14.4
77 Original Frankfurter grüne Soße 15 10.4
TotalPrice 1119.9
Page 1

Résultat de l'extraction : Après l'OCR (avec pytesseract ou EasyOCR), ou l'extraction avec PyPDF2 ou pdfplumber, le résultat obtenu est généralement une chaîne de texte brut, représentant les informations extraites du document, comme des mots, des phrases ou des données structurées, il nécessite des autres interventions pour le transformer et le structurer dans des bases de données bien optimisées -dites data warehouse-.

5. Data transformation :

Transformations par modèle NLP : Utilisation de modèles de traitement du langage naturel pour normaliser, extraire, et transformer le texte en informations structurées, permettant de mieux comprendre le contenu des documents.

Transformations par expressions régulières (Regex) : Application d'expressions régulières pour nettoyer, valider, et structurer les données extraites en supprimant les informations inutiles et en formatant correctement les résultats.

Remarque : les modèles NLP pré-entraînés sont très faibles si ne sont pas fine-tunés sur des tâches spécifiques, d'autre part ils permettent une transformation dynamique qui s'adapte avec tous les textes par contre les Regex qui sont quasiment statiques car ils cherchent sur des mots clés qui peuvent être changés chaque fois, ou bien posés dans un contexte différent.

1.5 Transformations par expressions régulières :

Exemple :

```
import re
import spacy

def parse_invoice_data(text):
    """
    Analyse et structure les données extraites de l'image d'une facture.
    :param text: Texte brut extrait de l'image.
    :return: Dictionnaire contenant les informations organisées.
    """
    data = {}

    # Extraction des informations générales
    data['Order ID'] = re.search(r'Order ID:\s*(\d+)', text).group(1)
    data['Customer ID'] = re.search(r'Customer ID:\s*(\w+)', text).group(1)
    data['Order Date'] = re.search(r'Order Date:\s*(\d-)+', text).group(1)

    # Extraction des détails du client
    customer_details = re.search(r'Customer Details:\n(.*)\nProduct Details:', text, re.S).group(1)
    details_lines = [line.split(':', 1) for line in customer_details.split('\n') if ':' in line]
    data['Customer Details'] = {line[0].strip(): line[1].strip() for line in details_lines}

    # Extraction des détails des produits
    product_details = re.findall(r'(\d+)\s+(.*)\s+(\d+)\s+([\d.]+)', text)
    data['Products'] = [
        {'Product ID': prod[0], 'Product Name': prod[1], 'Quantity': int(prod[2]), 'Unit Price': float(prod[3])}
        for prod in product_details
    ]

    # Extraction du prix total
    total_price = re.search(r'TotalPrice\s+([\d.]+)', text)
    data['Total Price'] = float(total_price.group(1)) if total_price else None

    return data
```

Résultat :(le code ci-dessus implémenter dans une model appelé nlp qui l'on va l'importer quand on aura le besoin...)

```
: from nlp import parse_invoice_data
processed_text1=parse_invoice_data(text3)
processed_text1

: {'Order ID': '10257',
  'Customer ID': 'HILAA',
  'Order Date': '2016-07-16',
  'Customer Details': {'Contact Name': 'Carlos Hernández',
  'Address': 'Carrera 22 con Ave. Carlos Soublette #8-35',
  'City': 'San Cristóbal',
  'Postal Code': '5022',
  'Country': 'Venezuela',
  'Phone': '(5) 555-1340',
  'Fax': '(5) 555-1948'},
  'Products': [{'Product ID': '27',
  'Product Name': 'Schoggi Schokolade',
  'Quantity': 25,
  'Unit Price': 35.1},
  {'Product ID': '39',
  'Product Name': 'Chartreuse verte',
  'Quantity': 6,
  'Unit Price': 14.4},
  {'Product ID': '77',
  'Product Name': 'Original Frankfurter grüne Soße',
  'Quantity': 15,
  'Unit Price': 10.4}],
  'Total Price': 1119.9}
```

2.5 Transformations par modèle NLP et Regex :

- Cette model nlp basé sur spacy normalement il n'arrive pas à extraire les informations correctement dans les tâches des entités recognitions car n'est pas fine_tuned a notre type de texte mais on va le combine avec les Regex pour des bonne résultat.

Example :

```
• import spacy
• import re
•
• # Charger le modèle spaCy
• nlp = spacy.load("fr_core_news_sm")
•
• def extract_invoice_data(text):
•     # Appliquer le NLP spaCy pour analyser le texte
•     doc = nlp(text)
•
•     # Extraire les données principales avec des expressions régulières
•     order_id = re.search(r"Order ID:\s*(\d+)", text).group(1)
•     customer_id = re.search(r"Customer ID:\s*(\w+)", text).group(1)
•     order_date = re.search(r"Order Date:\s*([\d-]+)", text).group(1)
•
•     # Extraire les détails du client (en utilisant NER de spaCy)
•     customer_details = {}
```

```

•     customer_details["Contact Name"] = None
•     customer_details["Address"] = None
•     customer_details["City"] = None
•     customer_details["Postal Code"] = None
•     customer_details["Country"] = None
•     customer_details["Phone"] = None
•     customer_details["Fax"] = None
•
•
•     for ent in doc.ents:
•         if ent.label_ == "PERSON" and customer_details["Contact Name"]
is None:
•             customer_details["Contact Name"] = ent.text
•             elif ent.label_ == "GPE" and customer_details["City"] is None:
•                 customer_details["City"] = ent.text
•             elif ent.label_ == "MONEY" and customer_details["Postal Code"]
is None:
•                 customer_details["Postal Code"] = ent.text
•             elif ent.label_ == "LOC" and customer_details["Country"] is
None:
•                 customer_details["Country"] = ent.text
•             elif "phone" in ent.text.lower() and customer_details["Phone"]
is None:
•                 customer_details["Phone"] = ent.text
•             elif "fax" in ent.text.lower() and customer_details["Fax"] is
None:
•                 customer_details["Fax"] = ent.text
•
•
•     # Extraire les produits
•     products_section = re.search(r"Product Details:(.*?)TotalPrice",
text, re.DOTALL).group(1)
•     product_lines = products_section.strip().split("\n")[1:] # Ignorer
la ligne d'en-tête
•     products = []
•     for line in product_lines:
•         match = re.match(r"(\d+)\s+(.*?)\s+(\d+)\s+([\d.]+)", line)
•         if match:
•             products.append({
•                 "Product ID": match.group(1),
•                 "Product Name": match.group(2).strip(),
•                 "Quantity": int(match.group(3)),
•                 "Unit Price": float(match.group(4)),
•             })
•
•
•     # Extraire le prix total
•     total_price = float(re.search(r"TotalPrice\s*([\d.]+)",
text).group(1))
•
•
•     # Construire le dictionnaire final
•     invoice_data = {

```



```

•         "Order ID": order_id,
•         "Customer ID": customer_id,
•         "Order Date": order_date,
•         "Customer Details": customer_details,
•         "Products": products,
•         "Total Price": total_price
•     }
•
•     return invoice_data
•

```

Résultat :

```

invoice_text = """
Invoice
Order ID: 10257
Customer ID: HILAA
Order Date: 2016-07-16
Customer Details:
Contact Name: Carlos Hernández
Address: Carrera 22 con Ave. Carlos Soublette #8-35
City: San Cristóbal
Postal Code: 5022
Country: Venezuela
Phone: (5) 555-1340
Fax: (5) 555-1948
Product Details:
Product ID Product Name Quantity Unit Price
27 Schoggi Schokolade 25 35.1
39 Chartreuse verte 6 14.4
77 Original Frankfurter grüne Soße 15 10.4
TotalPrice 1119.9
"""

# Appeler la fonction et afficher les résultats
invoice_data = extract_invoice_data(invoice_text)
print(invoice_data)

{'Order ID': '10257', 'Customer ID': 'HILAA', 'Order Date': '2016-07-16', 'Customer Details': {'Contact Name': None, 'Address': None, 'City': None, 'Postal Code': None, 'Country': 'San Cristóbal\nPostal', 'Phone': 'Venezuela\nPhone', 'Fax': 'Fax'}, 'Products': [{'Product ID': '27', 'Product Name': 'Schoggi Schokolade', 'Quantity': 25, 'Unit Price': 35.1}, {'Product ID': '39', 'Product Name': 'Chartreuse verte', 'Quantity': 6, 'Unit Price': 14.4}, {'Product ID': '77', 'Product Name': 'Original Frankfurter grüne Soße', 'Quantity': 15, 'Unit Price': 10.4}], 'Total Price': 1119.9}

```

Résultat de transformations : Les transformations par Regex et NLP permettent d'extraire et structurer des informations spécifiques (comme des dates, numéros, montants) à partir de textes non structurés. Cela rend les données prêtes pour un traitement ultérieur, comme l'analyse ou la visualisation.

6. Data loading :

Chargement des données dans un Data Warehouse: Les données transformées sont ensuite stockées dans un **Data Warehouse** pour des analyses efficaces. Les Data Warehouses centralisent les données provenant de différentes sources, facilitant ainsi l'analyse et la prise de décision.

Exemples de **Data Warehouses** :

- Amazon Redshift
- Google BigQuery
- Snowflake
- Apache Hive: Utilisé pour le traitement de données volumineuses dans un environnement Hadoop.

SQL est souvent utilisé pour interroger et charger des données dans le Data Warehouse, permettant des analyses rapides et efficaces.

Remarque : Pour notre situation nous allons structurer de manière définitive et télécharger les données dans un fichier Excel vu que notre objectif c'est de simuler le max le processus d'ETL pour le traitement de document.

Exemple :

```
import pandas as pd
import os

def save_invoice_to_excel(data, output_path):
    """
    Enregistre les données d'une facture dans un fichier Excel avec deux tables.
    Si le fichier existe déjà, il vérifie si l'Order ID existe déjà et ajoute les données sinon.
    :param data: Dictionnaire contenant les données structurées de la facture.
    :param output_path: Chemin du fichier Excel de sortie.
    """
    # Préparer les nouvelles données pour l'en-tête (Header)
    header_data = {
        "Order ID": [data['Order ID']],
        "Customer ID": [data['Customer ID']],
        "Order Date": [data['Order Date']],
        "Contact Name": [data['Customer Details'].get('Contact Name')],
        "Address": [data['Customer Details'].get('Address')],
        "City": [data['Customer Details'].get('City')],
        "Postal Code": [data['Customer Details'].get('Postal Code')],
        "Country": [data['Customer Details'].get('Country')],
        "Phone": [data['Customer Details'].get('Phone')],
        "Total Price": [data['Total Price']]
    }
    new_header_df = pd.DataFrame(header_data)

    # Préparer les nouvelles données pour les produits (Products)
    new_products_df = pd.DataFrame(data['Products'])
    new_products_df['Order ID'] = data['Order ID'] # Ajouter l'Order ID pour relier les produits à l'en-tête

    # Si le fichier Excel existe, charger les données existantes
    if os.path.exists(output_path):
        # Charger les feuilles existantes
        existing_header_df = pd.read_excel(output_path, sheet_name="Invoice Header")
        existing_products_df = pd.read_excel(output_path, sheet_name="Product Details")

        # Vérifier si l'Order ID existe déjà
        if data['Order ID'] in existing_header_df['Order ID'].values:
            print(f"L'Order ID {data['Order ID']} existe déjà dans le fichier. Aucune donnée ajoutée.")
            return

        # Ajouter les nouvelles données aux anciennes
        updated_header_df = pd.concat([existing_header_df, new_header_df], ignore_index=True)
        updated_products_df = pd.concat([existing_products_df, new_products_df], ignore_index=True)
    else:
        # Si le fichier n'existe pas, créer de nouvelles tables
        updated_header_df = new_header_df
        updated_products_df = new_products_df

    # Enregistrer les données mises à jour dans le fichier Excel
    with pd.ExcelWriter(output_path, engine='openpyxl', mode='w') as writer:
        updated_header_df.to_excel(writer, index=False, sheet_name="Invoice Header")
        updated_products_df.to_excel(writer, index=False, sheet_name="Product Details")

    print(f"Les données ont été enregistrées avec succès dans {output_path}.")
```

- Cette fonction permet de vérifier, unifier, formater ...et enregistrer les données

Résultat : (on va faire une appelle fonctions sur un texte déjà transformer plusieurs fois comme la suite)

```
[64]: from save import save_invoice_to_excel
output_path=r"C:\Users\dell\Desktop\BI_prototype\invoice_data.xlsx"
save_invoice_to_excel(processed_text1, output_path)
```

Les données ont été enregistrées avec succès dans C:\Users\dell\Desktop\BI_prototype\invoice_data.xlsx.

En réalise, un fichier Excel est créé, contient ensemble des donnees :

invoice_data.xlsx - E

File Home Insert Page Layout Formulas Data Review View Developer Help Tell me what you want to do

Paste Clipboard Font Alignment Number

GET GENUINE OFFICE Your license isn't genuine, and you may be a victim of software counterfeiting. Avoid interruption and keep your files safe

A1 Order ID

	A	B	C	D	E	F	G	H	I	J	K	L
1	Order ID	Customer ID	Order Date	Contact Name	Address	City	Postal Code	Country	Phone	Total Price		
2	10248	VINET	2016-07-01	Paul Henri	59 rue de la	Reims	51100	France	26.47.15.1	440		
3	10249	TOMSP	2016-07-01	Karin Josephs		Munster	44087	Germany	(0251-031-	1863.4		
4	10254	CHOPS	2016-07-11	'Yang Wan	Hauptstr. 2	Bern	3012	Switzerland	(0452.0765	625.2		
5	10258	ERNSH	2016-07-11	Roland Me	Kirchgasse 6		8010	Austria	7675-3425	2018.6		
6	10262	RATTC	2016-07-22		2817 Mito	'Albuquerque	87110	USA	(605) 555-4	624.8		
7	10265	BLONP	2016-07-2	Frédérique	24, place K	Strasbourg	67000	France	88.60.15.3	1176		
8	10268	GROSR	2016-07-3	Manuel Pe	5* Ave. Lo	Caracas	1081	Venezuela	(2) 283-29	1101.2		
9	10272	RATTC	2016-08-01	Paula Wils	2817 Milton Dr.		7110	usa	(608) 555-4	1456		
10	10280	BERGS	2016-08-1	Christina B	Berguvsvä	Lulea	'S-958 22	Sweden	(0921-12 3	613.2		
11	10257	HILAA	2016-07-1	Carlos Her	Carrera 22	San Cristó	5022	Venezuela	(5) 555-13-	1119.9		
12	10257	HILAA	2016-07-1	Carlos Her	Carrera 22	San Cristó	5022	Venezuela	(5) 555-13-	1119.9		
13	10257	HILAA	2016-07-1	Carlos Her	Carrera 22	San Cristó	5022	Venezuela	(5) 555-13-	1119.9		
14	10257	HILAA	2016-07-1	Carlos Her	Carrera 22	San Cristó	5022	Venezuela	(5) 555-13-	1119.9		
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												

Invoice Header Product Details (+)

A1					Product ID	
	A	B	C	D	E	F
1	Product ID	Product Name	Quantity	Unit Price	Order ID	
2	11	Queso Cab	12	14	10248	
3	42	Singaporea	10	9.8	10248	
4	72	Mozzarella	5	34.8	10248	
5	14	Tofu	9	18.6	10249	
6	51	Manjimup	40	42.4	10249	
7	24	Guarana Fa	15	3.6	10254	
8	55	Pate chino	21	19.2	10254	
9	2	Chang	50	152	10258	
10	5	Chef Antoi	65	170	10258	
11	32	Mascarpone	6	25.6	10258	
12	5	'Chef Anto	12	17	10262	
13	7	Uncle Bob	15	24	10262	
14	56	Gnocchi di	2	30.4	10262	
15	7	Alice Mutt	30	31.2	10265	
16	70	Outback La	20	12	10265	
17	29	Thiringer R	10	99	10268	
18	72	Mozzarella	4	27.8	10268	
19	20	Sit Rodney	6	64.8	10272	
20	31	Gorgonzol	40	10	10272	
21	72	Mozzarella	24	27.8	10272	
22	65	Fax: (0921	34	67	10280	
23	24	Guarana Fa	12	3.6	10280	
24	55	Paté chino	20	19.2	10280	
25	75	Rhonbrau	30	6.2	10280	

7. Data visualization :

Visualisation des données :

Utilisation d'une application interactive pour afficher et analyser les données traitées, permettant aux utilisateurs de consulter des graphiques, rapports et tendances à partir du Data Warehouse.

Outils de visualisation :

Intégration de bibliothèques comme **Plotly**, **Tableau** ou **Power BI** pour créer des visualisations dynamiques et interactives, offrant des vues détaillées et des filtres personnalisables.

Partie2 :Démonstration et simulation d'un processus BI :

1. Le scénario :

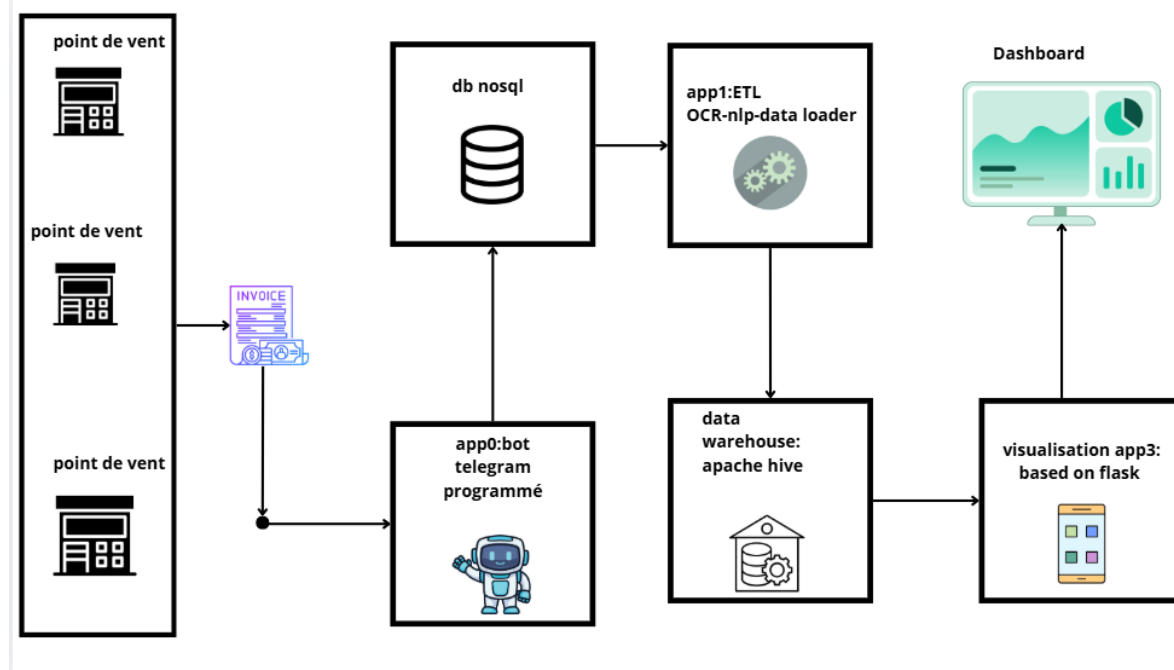
Nous sommes une entreprise spécialisée dans la vente de bons d'achat, avec 100 points de vente à travers le Maroc. Chaque point de vente génère des factures par un petit système local et les envoie à

notre système central via un bot Telegram, connecté à l'App1 qui ingère les documents en utilisant des requêtes CRUD. Les factures sont ensuite stockées temporairement.

Une seconde application (App2) effectue un processus ETL, utilisant un modèle OCR pour extraire le texte, et des modèles NLP ainsi que des expressions régulières pour transformer et structurer les données.

Enfin, les données traitées sont stockées dans un data warehouse et visualisées à travers une troisième application (App3), permettant leur consultation via un tableau de bord interactif.

2. L'architecture de système :



3. Explication de composant :

1.3 App0 : bot

Cette application définit un bot Telegram capable de recevoir et sauvegarder des fichiers multimédia tels que des images, audio, vidéos et PDF envoyés par l'ensemble des points de vente. Chaque point de vente envoie des fichiers via le bot, qui les télécharge, les stocke dans un répertoire local avec un nom unique incluant un timestamp, et répond à chaque point de vente pour confirmer la réception et l'enregistrement des fichiers.

```

boot.py > ...
1  import logging
2  import os
3  from datetime import datetime # Import to add timestamp
4  from telegram import Update, InputFile
5  from telegram.ext import Application, CommandHandler, MessageHandler, filters, CallbackContext
6
7  # Set up logging
8  logging.basicConfig(
9      format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
10     level=logging.INFO
11 )
12 logger = logging.getLogger(__name__)
13
14 # Your bot token
15 Token="jjjjjj" # Replace with your actual bot token
16
17 # Directory to save media files
18 MEDIA_DIR = r"C:\Users\dell\Desktop\BI_prototype\data"
19 if not os.path.exists(MEDIA_DIR):
20     os.makedirs(MEDIA_DIR)
21
22 # Function to download and save files
23 async def download_file(file_id: str, context: CallbackContext, file_name: str, extension: str):
24     """Download a file from Telegram and save it to the media directory."""
25     try:
26         # Add timestamp to make the filename unique
27         timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
28         unique_file_name = f"{file_name}_{timestamp}{extension}"
29
30         file = await context.bot.get_file(file_id)
31         file_path = os.path.join(MEDIA_DIR, unique_file_name)
32         await file.download_to_drive(file_path)
33         logger.info(f"Downloaded file: {file_path}")
34         return file_path
35     except Exception as e:
36         logger.error(f"Failed to download file: {e}")
37     return None

```

2.3 App1 :ETL basée en OCR, nlp et data loader :

- Ocr.py :Ce module utilise différentes bibliothèques OCR (Tesseract, PyPDF2, pdfplumber, et EasyOCR) pour implémenter ensemble des fonctions . Chaque fonction est dédiée à l'extraction de texte à partir de sources spécifiques, comme des images ou des pages PDF, et retourne le texte brut ou formaté extrait.


```

ocr.py > extract_with_pyteseract
1  import pytesseract
2  from PIL import Image
3  import pdfplumber
4  import easyocr
5  import PyPDF2
6  # Define path to tesseract executable if you did not add it to variable path
7
8  pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
9  # Open an image and extract text
10
11
12
13
14
15  def extract_with_pyteseract(image_path):
16      """
17      Utilise Tesseract OCR pour extraire du texte d'une image.
18      :param image_path: Chemin vers l'image.
19      :return: Texte brut extrait de l'image.
20      """
21      try:
22          # Ouvrir l'image
23          image = Image.open(image_path)
24          # Utiliser Tesseract pour extraire le texte
25          text = pytesseract.image_to_string(image)
26          return text
27      except Exception as e:
28          print(f"Erreur lors de l'extraction OCR : {e}")
29          return None
30
31
32
33
34
35
36  #3. Extraction de texte à partir de PDF (en utilisant pdfplumber)

```

- nlp.py : Ce module utilise spaCy et des expressions régulières pour extraire des informations structurées à partir du texte de factures, telles que l'ID de la commande, les détails du client, les produits et le prix total, retournant un dictionnaire organisé.

```

import re
import spacy

def parse_invoice_data(text):
    """
    Analyse et structure les données extraites de l'image d'une facture.
    :param text: Texte brut extrait de l'image.
    :return: Dictionnaire contenant les informations organisées.
    """
    data = {}

    # Extraction des informations générales
    data['Order ID'] = re.search(r'Order ID:\s*(\d+)', text).group(1)
    data['Customer ID'] = re.search(r'Customer ID:\s*(\w+)', text).group(1)
    data['Order Date'] = re.search(r'Order Date:\s*([\d-]+)', text).group(1)

    # Extraction des détails du client
    customer_details = re.search(r'Customer Details:\n(?:.|\n)*\nProduct Details:', text, re.S).group(1)
    details_lines = [line.split(':', 1) for line in customer_details.split('\n') if ':' in line]
    data['Customer Details'] = {line[0].strip(): line[1].strip() for line in details_lines}

    # Extraction des détails des produits
    product_details = re.findall(r'(\d+)\s+(?:.|\n)*\s+(\d+)\s+([\d.]+)', text)
    data['Products'] = [
        {'Product ID': prod[0], 'Product Name': prod[1], 'Quantity': int(prod[2]), 'Unit Price': float(prod[3])}
        for prod in product_details
    ]

    # Extraction du prix total
    total_price = re.search(r'TotalPrice\s+([\d.]+)', text)
    data['Total Price'] = float(total_price.group(1)) if total_price else None

    return data

```

- save.py : Ce module enregistre les données d'une facture dans un fichier Excel avec deux tables (en-tête et produits), en vérifiant si l'Order ID existe déjà avant d'ajouter les données ou de créer un nouveau fichier si nécessaire.

```

save.py > save_invoice_to_excel
1 import pandas as pd
2 import os
3
4 def save_invoice_to_excel(data, output_path):
5     """
6     Enregistre les données d'une facture dans un fichier Excel avec deux tables.
7     Si le fichier existe déjà, il vérifie si l'Order ID existe déjà et ajoute les données sinon.
8     :param data: Dictionnaire contenant les données structurées de la facture.
9     :param output_path: Chemin du fichier Excel de sortie.
10    """
11    # Préparer les nouvelles données pour l'en-tête (Header)
12    header_data = {
13        "Order ID": [data['Order ID']],
14        "Customer ID": [data['Customer ID']],
15        "Order Date": [data['Order Date']],
16        "Contact Name": [data['Customer Details'].get('Contact Name')],
17        "Address": [data['Customer Details'].get('Address')],
18        "City": [data['Customer Details'].get('City')],
19        "Postal Code": [data['Customer Details'].get('Postal Code')],
20        "Country": [data['Customer Details'].get('Country')],
21        "Phone": [data['Customer Details'].get('Phone')],
22        "Total Price": [data['Total Price']]
23    }
24    new_header_df = pd.DataFrame(header_data)
25
26    # Préparer les nouvelles données pour les produits (Products)
27    new_products_df = pd.DataFrame(data['Products'])
28    new_products_df['Order ID'] = data['Order ID'] # Ajouter l'Order ID pour relier les produits à l'en-tête
29
30    # Si le fichier Excel existe, charger les données existantes
31    if os.path.exists(output_path):
32        # Charger les feuilles existantes
33        existing_header_df = pd.read_excel(output_path, sheet_name="Invoice Header")
34        existing_products_df = pd.read_excel(output_path, sheet_name="Product Details")
35
36    # Concaténer les nouvelles données aux données existantes
37    existing_header_df = pd.concat([existing_header_df, new_header_df], ignore_index=True)
38    existing_products_df = pd.concat([existing_products_df, new_products_df], ignore_index=True)
39
40    # Sauvegarder les données dans un fichier Excel
41    existing_header_df.to_excel(output_path, sheet_name="Invoice Header", index=False)
42    existing_products_df.to_excel(output_path, sheet_name="Product Details", index=False)
43
44    return existing_header_df, existing_products_df

```

- App1.py : cette application combine les modules précédents pour extraire les données des factures à partir d'images et de PDF, les analyser avec NLP, puis les enregistrer dans un fichier Excel, tout en gérant les erreurs liées aux permissions

```

app1.py > ...
1  from ocr import extract_with_pytesesseract, extract_with_pdfplumber
2  from nlp import parse_invoice_data
3  from save import save_invoice_to_excel
4  import re
5  import pytesesseract
6  from PIL import Image
7  import spacy
8  import pandas as pd
9  import time
10 import os
11
12 processed = []
13 image_path = r"C:\Users\dell\Desktop\BI_prototype\data"
14 excel_path = r"C:\Users\dell\Desktop\BI_prototype\invoice_data.xlsx"
15
16 # Extensions des fichiers pris en charge
17 IMAGE_EXTENSIONS = {".jpg", ".jpeg", ".png", ".bmp", ".tiff"}
18 PDF_EXTENSION = ".pdf"
19
20 while True:
21     # Lister tous les fichiers dans le dossier
22     files = os.listdir(image_path)
23     files = [os.path.join(image_path, file) for file in files]
24
25     for file in files:
26         if file not in processed:
27             try:
28                 # Déterminer le type de fichier

```

3.3 App2 : la visualisation

- Vis.py : Cette application combine Flask et Dash pour afficher des graphiques et des résumés à partir d'un fichier Excel. Elle nettoie les données, affiche le nombre de demandes et le prix total par pays, et met à jour les informations toutes les 5 secondes. Les utilisateurs peuvent filtrer par pays et consulter les cartes de produits les plus vendus et rentables.

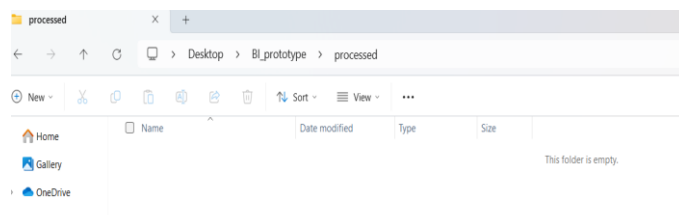
```

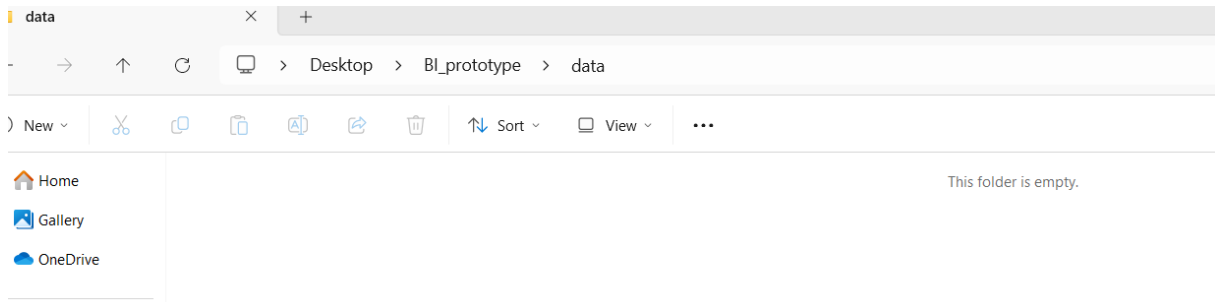
vis.py > ...
1  from flask import Flask
2  import dash
3  from dash import dcc, html
4  from dash.dependencies import Output, Input
5  import pandas as pd
6  import os
7  import time
8
9  # Chemin vers le fichier Excel
10 FILE_PATH = "invoice_data.xlsx" # Assurez-vous que ce fichier existe
11
12 # Initialisation de Flask
13 server = Flask(__name__)
14
15 # Intégration de Dash à Flask
16 app = dash.Dash(__name__, server=server, url_base_pathname='/dashboard/')
17
18 # Fonction pour nettoyer les données
19 def clean_data(df):
20     df.rename(columns=lambda x: x.strip(), inplace=True)
21     df['Country'] = df['Country'].str.replace('[\s]', '').str.strip()
22     df['Country'] = df['Country'].str.replace('Italy', 'Italy')
23     df['Total Price'] = pd.to_numeric(df['Total Price'], errors='coerce')
24     return df
25
26 # Fonction pour lire et traiter les données de la première feuille
27 def read_excel_data(file_path):
28     try:
29         df = pd.read_excel(file_path, sheet_name=0, engine='openpyxl')
30         df = clean_data(df) # Nettoyer les données
31
32         if 'Country' in df.columns and 'Total Price' in df.columns:
33             grouped = df.groupby('Country').agg({
34                 'Country': 'count', # Nombre de demandes
35                 'Total Price': 'sum' # Somme des prix totaux

```

4. L'exécutions :

Etat 0 : aucun file Excel ((supposé une data warehouse), aucune donnée dans le dossier data (supposé un dB nosql) :





Etat1 : on va lance app0 ,app1 et app3 :

Pas de fichier Excel , pas de données en dossier data.

```
(BI) C:\Users\dell\Desktop\BI_prototype>python boot.py
024-11-30 03:51:38,625 - httpx - INFO - HTTP Request: POST https://api.telegram.org/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/getMe "HTTP/1.1 200 OK"
024-11-30 03:51:38,683 - httpx - INFO - HTTP Request: POST https://api.telegram.org/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/deleteWebhook "HTTP/1.1 200 OK"
024-11-30 03:51:38,687 - telegram.ext.Application - INFO - Application started
```

```
(BI) C:\Users\dell\Desktop\BI_prototype>python app1.py
```

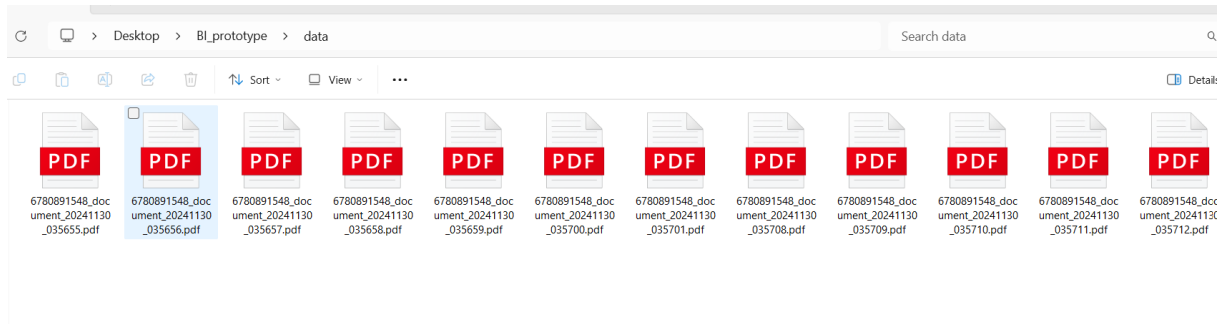
```
(BI) C:\Users\dell\Desktop\BI_prototype>python vis.py
Erreur lors de la lecture des données : [Errno 2] No such file or directory: 'C:\\Users\\dell\\Desktop\\BI_prototype\\processed\\invoice_data.xlsx'
Le fichier spécifié n'existe pas. Réessaie dans 5 secondes...
Le fichier spécifié n'existe pas. Réessaie dans 5 secondes...
Le fichier spécifié n'existe pas. Réessaie dans 5 secondes...
```

Etat2 : on va envoyer des file.pdf et des images d'ensemble des factures via télégramme :

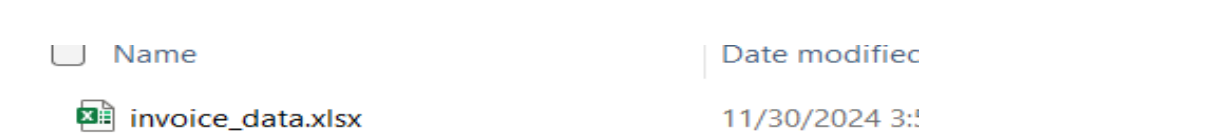


```
024-11-30 03:57:11,117 - httpx - INFO - HTTP Request: POST https://api.telegram.org/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/sendMessage "HTTP/1.1 200 OK"
024-11-30 03:57:11,170 - httpx - INFO - HTTP Request: POST https://api.telegram.org/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/getFile "HTTP/1.1 200 OK"
024-11-30 03:57:11,257 - httpx - INFO - HTTP Request: GET https://api.telegram.org/file/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/documents/file_154.pdf "HTTP/1.1 200 OK"
024-11-30 03:57:11,259 - _main_ - INFO - Downloaded file: C:\Users\dell\Desktop\BI_prototype\data\6780891548_document_20241130_035711.pdf
024-11-30 03:57:11,332 - httpx - INFO - HTTP Request: POST https://api.telegram.org/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/sendMessage "HTTP/1.1 200 OK"
024-11-30 03:57:11,396 - httpx - INFO - HTTP Request: POST https://api.telegram.org/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/getFile "HTTP/1.1 200 OK"
024-11-30 03:57:11,478 - httpx - INFO - HTTP Request: GET https://api.telegram.org/file/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/documents/file_155.pdf "HTTP/1.1 200 OK"
024-11-30 03:57:11,480 - _main_ - INFO - Downloaded file: C:\Users\dell\Desktop\BI_prototype\data\6780891548_document_20241130_035711.pdf
024-11-30 03:57:11,587 - httpx - INFO - HTTP Request: POST https://api.telegram.org/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/sendMessage "HTTP/1.1 200 OK"
024-11-30 03:57:11,655 - httpx - INFO - HTTP Request: GET https://api.telegram.org/file/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/documents/file_156.pdf "HTTP/1.1 200 OK"
024-11-30 03:57:11,712 - httpx - INFO - HTTP Request: GET https://api.telegram.org/file/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/documents/file_156.pdf "HTTP/1.1 200 OK"
024-11-30 03:57:11,712 - _main_ - INFO - Downloaded file: C:\Users\dell\Desktop\BI_prototype\data\6780891548_document_20241130_035711.pdf
024-11-30 03:57:11,788 - httpx - INFO - HTTP Request: POST https://api.telegram.org/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/sendMessage "HTTP/1.1 200 OK"
024-11-30 03:57:11,863 - httpx - INFO - HTTP Request: GET https://api.telegram.org/file/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/documents/file_157.pdf "HTTP/1.1 200 OK"
024-11-30 03:57:11,922 - httpx - INFO - HTTP Request: GET https://api.telegram.org/file/bot7812133141:AAEqs6ncM_oBYgegx2IvAP5vDuZDP2KMIVc/documents/file_157.pdf "HTTP/1.1 200 OK"
024-11-30 03:57:11,922 - _main_ - INFO - Downloaded file: C:\Users\dell\Desktop\BI_prototype\data\6780891548_document_20241130_035711.pdf
```


- Les données ont enregistré dans le dossier data :



- File Excel été créé :



Order ID	Customer ID	Order Date	Contact Name	Address	City	Postal Code	Country	Phone	Total Price		
10840	LINOD	2018-01-1	Felipe Izqu	Ave. 5 de M. de Marg		4980	Venezuela	(8) 34-56-1	264		
10842	TORTU	2018-01-2	Miguel Ang	Avda. Azte	México D.	5033	Mexico	(5) 555-29	975		
10844	PICCO	2018-01-2	Georg Pipp	Geislweg 1	Salzburg	5020	Austria	6562-9722	735		
10849	KOENE	2018-01-2	Philip Cran	Maubelstr.	Brandenbu	14776	Germany	0555-0987	1052.14		
10853	BLAUS	2018-01-2	Hanna Mo	Forsterstr.	Mannheim	68306	Germany	0621-0846	625		
10856	ANTON	2018-01-2	Antonio M	Mataderos	México D.	5023	Mexico	(5) 555-39	660		
10859	FRANK	2018-01-2	Peter Fran	Berliner Pl	München	80805	Germany	089-08773	1438.25		
10860	FRANR	2018-01-2	Carine Sch	54, rue Ro	Nantes	44000	France	40.32.21.2	519		
10863	HILAA	2018-02-0	Carlos Her	Carrera 22	San Cristó	5022	Venezuela	(5) 555-13	519		
10866	BERGS	2018-02-0	Christina B	Berguvsvä	Luleå	S-958 22	Sweden	0921-12 34	1461.6		
10870	WOLZA	2018-02-0	Zbyszek Pi	ul. Filtrow	Warszawa	01-012	Poland	(26) 642-7	160		
10874	GODOS	2018-02-0	José Pedrc	C/ Romero	Sevilla	41101	Spain	(95) 555 8	310		

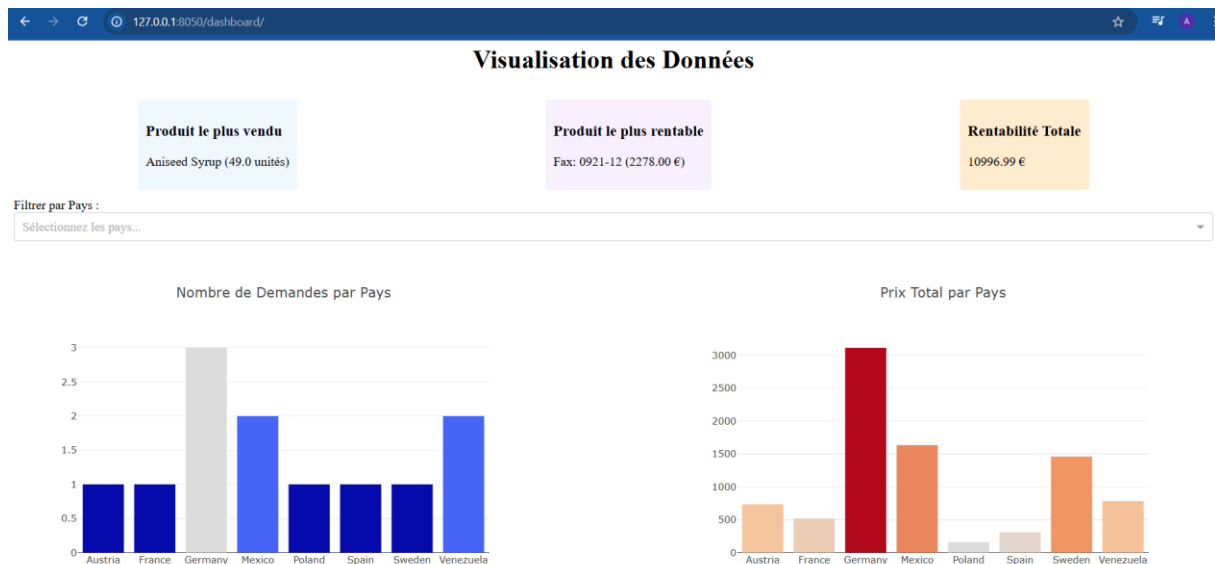
- Finalement l'app2 de visualization arrive a trouver le file et été lance aussi :

```

Le fichier spécifié n'existe pas. Réessaie dans 5 secondes...
Le fichier spécifié n'existe pas. Réessaie dans 5 secondes...
Le fichier spécifié n'existe pas. Réessaie dans 5 secondes...
Le fichier spécifié n'existe pas. Réessaie dans 5 secondes...
Le fichier spécifié n'existe pas. Réessaie dans 5 secondes...
Le fichier spécifié n'existe pas. Réessaie dans 5 secondes...
Le fichier spécifié n'existe pas. Réessaie dans 5 secondes...
Le fichier spécifié n'existe pas. Réessaie dans 5 secondes...
Fichier trouvé. Lancement du serveur...
* Serving Flask app 'vis'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8050
* Running on http://192.168.11.104:8050
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
Fichier trouvé. Lancement du serveur...
* Debugger is active!
* Debugger PIN: 128-175-003

```

La visualisation bien générer :



Etat3 : on va envoyer des images au lieu du PDF maintenant juste on va vérifier la mise a jour de Dashboard :

Country: Finland

Phone: 981-443655

Fax: 981-443655

Product Details:

Product ID	Product Name	Quantity	Unit Price
49	Biscuits Crisps Malt	50	14.2
50	Biscuits Crisps Malt	40	44.0

Customer ID: C0058

Order Date: 2019-07-09

Customer Details:

Customer Name:	Manus Pater
Address:	17 Rue de la République
City:	Paris
Postal Code:	75001
Country:	France
Phone:	01 23 45 67 89
Fax:	01 23 45 67 89

Product Details:

Product ID	Product Name	Quantity	Unit Price
49	Biscuits Crisps Malt	50	14.2
50	Biscuits Crisps Malt	40	44.0

Image received and saved successfully! 04:06

Image received and saved successfully! 04:06

Image received and saved successfully! 04:06

Image received and saved successfully! 04:06

- Le Dashboard n'été pas mise a jour ,et L'app1 demande le fermeture de file Excel pour être capable d'écrire les données :

```
Nouvelles données ajoutées avec succès.
Traitement du PDF : C:\Users\dell\Desktop\BI_prototype\data\6780891548_document_20241130_035712.pdf
Les données ont été enregistrées avec succès dans C:\Users\dell\Desktop\BI_prototype\processed\invoice_data.xlsx.
Nouvelles données ajoutées avec succès.
Traitement de l'image : C:\Users\dell\Desktop\BI_prototype\data\6780891548_image_20241130_040618.jpg
Le fichier C:\Users\dell\Desktop\BI_prototype\processed\invoice_data.xlsx est ouvert. Veuillez le fermer pour continuer.
Le fichier C:\Users\dell\Desktop\BI_prototype\processed\invoice_data.xlsx est ouvert. Veuillez le fermer pour continuer.
Le fichier C:\Users\dell\Desktop\BI_prototype\processed\invoice_data.xlsx est ouvert. Veuillez le fermer pour continuer.
Le fichier C:\Users\dell\Desktop\BI_prototype\processed\invoice_data.xlsx est ouvert. Veuillez le fermer pour continuer.
Le fichier C:\Users\dell\Desktop\BI_prototype\processed\invoice_data.xlsx est ouvert. Veuillez le fermer pour continuer.
```

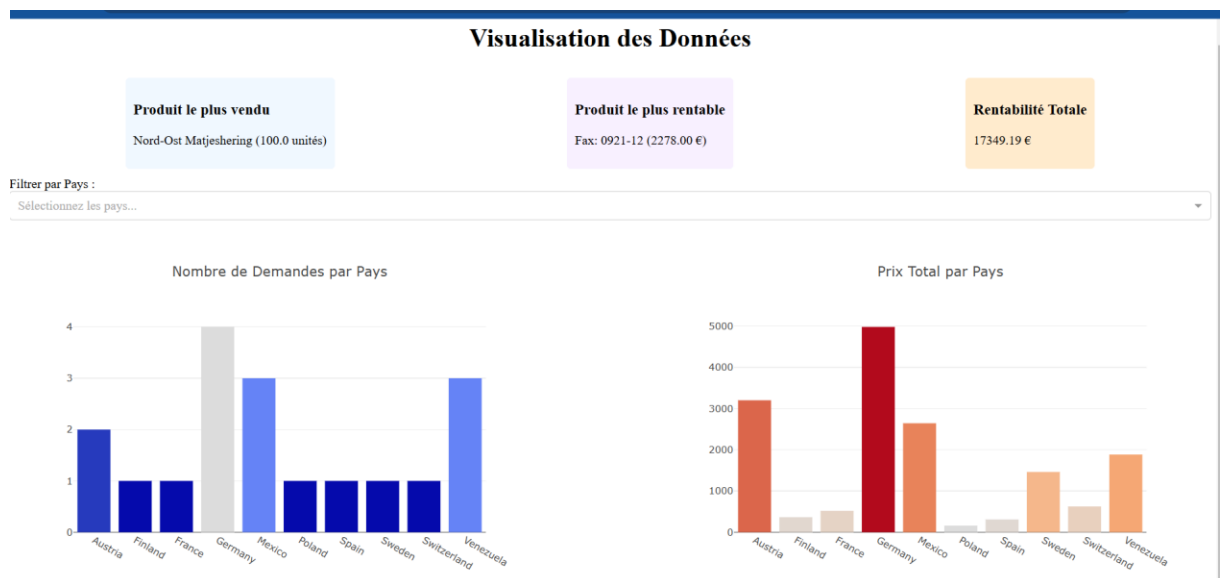
- On ferme le fichier :

```

Le fichier C:\Users\dell\Desktop\BI_prototype\processed\invoice_data.xlsx est ouvert. Veuillez le fermer pour continuer.
Le fichier C:\Users\dell\Desktop\BI_prototype\processed\invoice_data.xlsx est ouvert. Veuillez le fermer pour continuer.
Les données ont été enregistrées avec succès dans C:\Users\dell\Desktop\BI_prototype\processed\invoice_data.xlsx.
Nouvelles données ajoutées avec succès.
Traitement de l'image : C:\Users\dell\Desktop\BI_prototype\data\6780891548_image_20241130_040619.jpg
Les données ont été enregistrées avec succès dans C:\Users\dell\Desktop\BI_prototype\processed\invoice_data.xlsx.
Nouvelles données ajoutées avec succès.
Traitement de l'image : C:\Users\dell\Desktop\BI_prototype\data\6780891548_image_20241130_040620.jpg
Les données ont été enregistrées avec succès dans C:\Users\dell\Desktop\BI_prototype\processed\invoice_data.xlsx.
Nouvelles données ajoutées avec succès.

```

- Le Dashboard est met a jour avec succès



- Comme ça nous réalisons un Solutions BI quasiment complet vue que nous n'avons pas utilisé un data warehouse

Conclusion générale :

Dans le cadre de cette application, nous avons vu comment l'intégration de plusieurs modules pour l'extraction, le traitement, et la visualisation des données peut s'apparenter à une solution de **Business Intelligence (BI)**. La **BI** vise à transformer les données brutes en informations exploitables, facilitant ainsi la prise de décisions stratégiques dans les entreprises.

Ici, nous avons appliqué des principes de BI en utilisant des outils comme **Dash** et **Flask** pour créer une interface interactive, permettant aux utilisateurs de filtrer et d'explorer les données de manière dynamique. La visualisation des informations sous forme de graphiques, en temps réel, fournit des perspectives clés sur les tendances de ventes et les performances des produits, tout en permettant une prise de décision plus rapide et éclairée.

Ainsi, cette approche illustre comment une application BI peut non seulement centraliser et traiter les données, mais aussi les rendre accessibles et compréhensibles pour les utilisateurs non techniques, améliorant ainsi la productivité et l'efficacité dans des contextes d'affaires. En fin de compte, le but de la BI est d'offrir une vision claire des données pour soutenir des décisions informées et basées sur des faits concrets.