PDF DATA MASKING

TEAM NO : 06

TEAM NAME: TECH SPARTANS

**Objective:**

Develop a robust application that masks sensitive information from input PDFs, ensuring data privacy and security. The application must handle text and images within the PDF, accommodating diverse names across different demographics, including Malaysian, Chinese, and Korean names.

**PROBLEM STATEMENT:**

Developing a secure application to mask sensitive information in PDFs presents a challenge in ensuring comprehensive data privacy. The application must be capable of accurately identifying and obscuring sensitive data within both text and images. It must handle names from diverse demographics, including Malaysian, Chinese, and Korean, to prevent unauthorized access to personal information. The solution requires a robust backend for processing and masking data, integrated with a user-friendly frontend for efficient interaction and usability.

**SOURCE CODE:**

**(FILE NAME)**

pdf\_masking.py**:**

import fitz # PyMuPDF

import spacy

import phonenumbers

import re

import tkinter as tk

from tkinter import filedialog, messagebox

# Load the custom spaCy model

try:

nlp = spacy.load('custom\_ner\_model')

except OSError:

nlp = spacy.load('en\_core\_web\_sm')

def extract\_text\_and\_positions\_from\_pdf(pdf\_path):

doc = fitz.open(pdf\_path)

text\_and\_positions = []

for page\_num, page in enumerate(doc):

blocks = page.get\_text("blocks")

for b in blocks:

block\_text = b[4]

rect = fitz.Rect(b[:4])

text\_and\_positions.append((block\_text, rect, page\_num))

return text\_and\_positions, doc

def mask\_sensitive\_information(text\_and\_positions, doc):

sensitive\_info = []

for text, rect, page\_num in text\_and\_positions:

doc\_page = nlp(text)

for ent in doc\_page.ents:

if ent.label\_ in ['PERSON', 'ORG', 'GPE', 'LOC', 'EMAIL', 'PHONE\_NUMBER']:

sensitive\_info.append((ent.text, rect, page\_num))

# Detect phone numbers using phonenumbers library

phone\_numbers = phonenumbers.PhoneNumberMatcher(text, "US")

for match in phone\_numbers:

phone\_number = text[match.start:match.end]

sensitive\_info.append((phone\_number, rect, page\_num))

# Detect emails using regex

email\_pattern = r'[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+'

emails = re.findall(email\_pattern, text)

for email in emails:

sensitive\_info.append((email, rect, page\_num))

# Detect "Name:" and "Phone Number:" fields

name\_pattern = r'Name\s\*:\s\*\w+'

names = re.findall(name\_pattern, text)

for name in names:

sensitive\_info.append((name, rect, page\_num))

phone\_pattern = r'Phone\s\*Number\s\*:\s\*\d+|Ph\s\*no\s\*:\s\*\d+'

phones = re.findall(phone\_pattern, text)

for phone in phones:

sensitive\_info.append((phone, rect, page\_num))

for sensitive\_text, rect, page\_num in sensitive\_info:

page = doc.load\_page(page\_num)

page.add\_redact\_annot(rect, text='', fill=(0, 0, 0))

page.apply\_redactions()

def select\_pdf():

file\_path = filedialog.askopenfilename(filetypes=[("PDF files", "\*.pdf")])

if file\_path:

entry\_pdf\_path.delete(0, tk.END)

entry\_pdf\_path.insert(0, file\_path)

def process\_pdf():

pdf\_path = entry\_pdf\_path.get()

if not pdf\_path:

messagebox.showerror("Error", "Please select a PDF file")

return

text\_and\_positions, doc = extract\_text\_and\_positions\_from\_pdf(pdf\_path)

mask\_sensitive\_information(text\_and\_positions, doc)

output\_pdf\_path = filedialog.asksaveasfilename(defaultextension=".pdf", filetypes=[("PDF files", "\*.pdf")])

if output\_pdf\_path:

doc.save(output\_pdf\_path)

messagebox.showinfo("Success", "Masked PDF created successfully!")

# Create GUI

root = tk.Tk()

root.title("PDF Masking Tool By Sriram ")

frame = tk.Frame(root, padx=10, pady=10)

frame.pack(padx=10, pady=10)

label\_pdf\_path = tk.Label(frame, text="PDF File:")

label\_pdf\_path.grid(row=0, column=0, sticky=tk.W)

entry\_pdf\_path = tk.Entry(frame, width=50)

entry\_pdf\_path.grid(row=0, column=1, padx=5, pady=5)

button\_browse = tk.Button(frame, text="Browse...", command=select\_pdf)

button\_browse.grid(row=0, column=2, padx=5, pady=5)

button\_process = tk.Button(frame, text="Process PDF", command=process\_pdf)

button\_process.grid(row=1, column=0, columnspan=3, pady=10)

root.mainloop()

**(FILE NAME )**

train\_ner.py:

import spacy

from spacy.training import Example

from training\_data import TRAIN\_DATA

import random

def train\_spacy(data, iterations):

nlp = spacy.blank("en") # Create a blank language class

if "ner" not in nlp.pipe\_names:

ner = nlp.add\_pipe("ner", last=True)

else:

ner = nlp.get\_pipe("ner")

for \_, annotations in data:

for ent in annotations.get("entities"):

ner.add\_label(ent[2])

optimizer = nlp.begin\_training()

for itn in range(iterations):

print(f"Starting iteration {itn}")

random.shuffle(data)

losses = {}

for text, annotations in data:

example = Example.from\_dict(nlp.make\_doc(text), annotations)

nlp.update([example], drop=0.5, losses=losses)

print(losses)

return nlp

# Train the model

nlp = train\_spacy(TRAIN\_DATA, 30)

# Save the model

nlp.to\_disk("custom\_ner\_model")

**(File name)**

training\_data.py:

TRAIN\_DATA = [

("John Doe's email is john.doe@example.com", {"entities": [(0, 8, "PERSON"), (22, 40, "EMAIL")]}),

("Call me at (123) 456-7890", {"entities": [(11, 25, "PHONE\_NUMBER")]}),

("Name: Alice, Phone Number: 9876543210", {"entities": [(6, 11, "PERSON"), (26, 36, "PHONE\_NUMBER")]}),

("Ph no: 1234567890, Name: Bob", {"entities": [(6, 16, "PHONE\_NUMBER"), (24, 27, "PERSON")]}),

("Contact me at bob@example.com or (555) 123-4567", {"entities": [(13, 29, "EMAIL"), (33, 47, "PHONE\_NUMBER")]}),

("Sriram's number is 9876543210", {"entities": [(0, 6, "PERSON"), (20, 30, "PHONE\_NUMBER")]}),

# Indian names and numbers

("Name: Aarav, Phone Number: 9876543210", {"entities": [(6, 11, "PERSON"), (26, 36, "PHONE\_NUMBER")]}),

("Contact: Priya Sharma, Ph no: 9876543210", {"entities": [(10, 22, "PERSON"), (34, 44, "PHONE\_NUMBER")]}),

("Ravi Kumar's email: ravi.kumar@domain.in", {"entities": [(0, 10, "PERSON"), (17, 35, "EMAIL")]}),

("Reach out to Ananya at ananya@example.in or call 1234567890",

{"entities": [(12, 18, "PERSON"), (22, 40, "EMAIL"), (46, 56, "PHONE\_NUMBER")]}),

("Nikhil's number: 9876543210, Name: Sneha",

{"entities": [(0, 6, "PERSON"), (17, 27, "PHONE\_NUMBER"), (31, 36, "PERSON")]}),

("Ravi Singh, phone: 8765432109", {"entities": [(0, 10, "PERSON"), (14, 24, "PHONE\_NUMBER")]}),

("Name: Neha Gupta, Ph No: 9999999999", {"entities": [(6, 16, "PERSON"), (22, 32, "PHONE\_NUMBER")]}),

("Mr. Rajesh Kumar's contact: 8888888888", {"entities": [(4, 15, "PERSON"), (22, 32, "PHONE\_NUMBER")]}),

("Name: Sanjay Patel, Phone Number: 9876543210", {"entities": [(6, 16, "PERSON"), (31, 41, "PHONE\_NUMBER")]}),

("Email: amit@xyz.in, Contact: Rina Sharma, Ph no: 8888888888",

{"entities": [(6, 19, "EMAIL"), (26, 36, "PERSON"), (42, 52, "PHONE\_NUMBER")]}),

# More Indian names and numbers

("Name: Aarav Patel, Phone Number: 9876543210", {"entities": [(6, 16, "PERSON"), (31, 41, "PHONE\_NUMBER")]}),

("Contact person: Meera Singh, Phone No: 9123456789", {"entities": [(16, 26, "PERSON"), (32, 42, "PHONE\_NUMBER")]}),

("Raj Kumar's email: raj.kumar@abc.com", {"entities": [(0, 10, "PERSON"), (17, 34, "EMAIL")]}),

("Ritika, phone: 8777665544", {"entities": [(0, 6, "PERSON"), (13, 23, "PHONE\_NUMBER")]}),

("Phone Number: 9998887777, Name: Vikas", {"entities": [(14, 24, "PHONE\_NUMBER"), (30, 35, "PERSON")]}),

("Name: Divya Singh, Phone Number: 7654321098", {"entities": [(6, 16, "PERSON"), (31, 41, "PHONE\_NUMBER")]}),

("Contact: Ashok Kumar, Phone: 9823456789", {"entities": [(9, 19, "PERSON"), (26, 36, "PHONE\_NUMBER")]}),

("Mr. Sumit Sharma, Phone No: 9999999999", {"entities": [(3, 14, "PERSON"), (22, 32, "PHONE\_NUMBER")]}),

("Email: priyanka@gmail.com, Contact: Alok, Ph no: 9765432109",

{"entities": [(6, 25, "EMAIL"), (33, 37, "PERSON"), (42, 52, "PHONE\_NUMBER")]}),

("Phone Number: 8887776666, Name: Kiran", {"entities": [(14, 24, "PHONE\_NUMBER"), (30, 35, "PERSON")]}),

# Adding more Indian examples

("Name: Rahul, Phone Number: 9876543210", {"entities": [(6, 11, "PERSON"), (26, 36, "PHONE\_NUMBER")]}),

("Rajesh's contact: 9823456789", {"entities": [(0, 6, "PERSON"), (15, 25, "PHONE\_NUMBER")]}),

("Sonia Patel, Phone: 9998887777", {"entities": [(0, 11, "PERSON"), (18, 28, "PHONE\_NUMBER")]}),

("Contact: Deepak Sharma, Phone No: 8889997777", {"entities": [(9, 21, "PERSON"), (27, 37, "PHONE\_NUMBER")]}),

("Name: Priyanka, Phone Number: 9876543210", {"entities": [(6, 15, "PERSON"), (30, 40, "PHONE\_NUMBER")]}),

("Mr. Anil Kumar, Phone No: 9999999999", {"entities": [(3, 13, "PERSON"), (21, 31, "PHONE\_NUMBER")]}),

("Shivani's phone number is 8765432109", {"entities": [(0, 8, "PERSON"), (24, 34, "PHONE\_NUMBER")]}),

("Name: Neha, Ph No: 9900112233", {"entities": [(6, 10, "PERSON"), (15, 25, "PHONE\_NUMBER")]}),

("Contact person: Alok, Phone: 9999999999", {"entities": [(15, 19, "PERSON"), (26, 36, "PHONE\_NUMBER")]}),

("Name: Priyanka, Phone Number: 9998887777", {"entities": [(6, 15, "PERSON"), (30, 40, "PHONE\_NUMBER")]}),

# Adding further examples

("Phone: 1234567890, Name: Neelam", {"entities": [(0, 10, "PHONE\_NUMBER"), (15, 20, "PERSON")]}),

("Name: Vikram Singh, Ph no: 9997778888", {"entities": [(6, 17, "PERSON"), (22, 32, "PHONE\_NUMBER")]}),

("Email: rakesh@domain.co.in, Contact: Sunita, Phone Number: 9823456789",

{"entities": [(6, 27, "EMAIL"), (36, 42, "PERSON"), (48, 58, "PHONE\_NUMBER")]}),

("Sumit Sharma's phone number is 9712345678", {"entities": [(0, 11, "PERSON"), (27, 37, "PHONE\_NUMBER")]}),

("Name: Aarav, Phone Number: 9112233445", {"entities": [(6, 11, "PERSON"), (26, 36, "PHONE\_NUMBER")]}),

("Contact: Anju, Ph no: 9991112222", {"entities": [(9, 13, "PERSON"), (19, 29, "PHONE\_NUMBER")]}),

("Name: Manish, Phone Number: 8887776666", {"entities": [(6, 12, "PERSON"), (27, 37, "PHONE\_NUMBER")]}),

("Mr. Rajiv Kumar, Phone No: 9911223344", {"entities": [(3, 13, "PERSON"), (22, 32, "PHONE\_NUMBER")]}),

("Phone Number: 9812345678, Name: Riya", {"entities": [(14, 24, "PHONE\_NUMBER"), (30, 34, "PERSON")]}),

("Name: Ramesh, Contact: 9998887777", {"entities": [(6, 12, "PERSON"), (21, 31, "PHONE\_NUMBER")]}),

("Email: jaya@example.in, Name: Priya, Phone Number: 9000112233",

{"entities": [(6, 22, "EMAIL"), (24, 29, "PERSON"), (35, 45, "PHONE\_NUMBER")]}),

# Adding even more examples

("Phone No: 9700012345, Name: Sandeep", {"entities": [(9, 19, "PHONE\_NUMBER"), (25, 31, "PERSON")]}),

("Contact Person: Aman, Phone Number: 9556778899", {"entities": [(15, 19, "PERSON"), (30, 40, "PHONE\_NUMBER")]}),

("Name: Clark Carson, Phone Number: 9823456789", {"entities": [(6, 11, "PERSON"), (26, 36, "PHONE\_NUMBER")]}),

("Contact: Sunil, Ph No: 9911223344", {"entities": [(9, 14, "PERSON"), (19, 29, "PHONE\_NUMBER")]}),

("Mr. Arjun's email is arjun@example.in", {"entities": [(3, 8, "PERSON"), (16, 33, "EMAIL")]}),

("Name: SRIRAM, Phone Number: 9865432109", {"entities": [(6, 11, "PERSON"), (26, 36, "PHONE\_NUMBER")]}),

("Ravi's contact number: 9999999999", {"entities": [(0, 4, "PERSON"), (21, 31, "PHONE\_NUMBER")]}),

("Phone Number: 9777888999, Name: Lata", {"entities": [(14, 24, "PHONE\_NUMBER"), (30, 34, "PERSON")]}),

("Name: GITA, Phone: 9911223344", {"entities": [(6, 11, "PERSON"), (17, 27, "PHONE\_NUMBER")]}),

("Contact: Raj, Phone No: 9823456789", {"entities": [(9, 12, "PERSON"), (19, 29, "PHONE\_NUMBER")]}),

]

**USER GUIDE:**

**STEP 1:**

**INSTALL LIBRARIES**

* pip install PyMuPDF
* pip install spacy
* pip install phonenumbers
* pip install tk
* python -m spacy download en\_core\_web\_sm
* pip install spacy
* pip install spacy-transformers
* pip install pandas # For handling datasets
* pip install spacy spacy-transformers pandas PyMuPDF phonenumbers tk
* python -m spacy download en\_core\_web\_sm

STEP 2:

* After installing the libraries (have to install every library) .
* Have to name the file name exactly given in this document.
* Have to write the given code in the given file names.
* Go to terminal and run “python pdf\_masking.py” or run the project.

**STEP 3:**

* Select pdf that needs to be masked and click process pdf button, after masking the pdf then you can have the masked output pdf downloaded where ever you want to download

**RECOMMENDED COMPILER:**

Pycharm professional

**THANK YOU**

CHEMUKULA THULASI SANDEEP:- RA2211003011377

K.LEELA SRIRAM:- RA2211003011385