import fitz # PyMuPDF import re import nltk import pandas as pd import spacy from flair.data import Sentence from flair.models import SequenceTagger from collections import Counter from nltk.corpus import stopwords from huggingface\_hub import login # Introduce tu token aquí token = "xxxxxxxxxxxx" login(token=token) # Ahora puedes cargar el modelo # The model name has been changed from 'ner-spanish' to 'flair/ner-spanish-large' tagger = SequenceTagger.load('flair/ner-spanish-large') pytorch\_model.bin: 0%| | 0.00/2.24G [00:00<?, ?B/s] tokenizer\_config.json: 0%| | 0.00/25.0 [00:00<?, ?B/s] config.json: 0%| | 0.00/616 [00:00<?, ?B/s] sentencepiece.bpe.model: 0%| | 0.00/5.07M [00:00<?, ?B/s] tokenizer.json: 0%| | 0.00/9.10M [00:00<?, ?B/s] 2025-02-14 16:33:23,271 SequenceTagger predicts: Dictionary with 20 tags: <unk>, O, S-LOC, S-ORG, B-PER, I-PER, E-PER, S-MISC, B-ORG, E-ORG, S-PER, I-ORG, B-LOC, E-LOC, B-MISC, E-MISC, I-MISC, I-LOC, <START>, <STOP> # Ruta del archivo PDF pdf\_path = '/content/certificado (2).pdf' # 1. Leer y unificar el texto def read\_pdf(file\_path): doc = fitz.open(file\_path) text = '' for page in doc: text += page.get\_text() doc.close() return text # 2. Preprocesamiento del texto def preprocess\_text(text): # Unificar Unicode text = text.encode('utf-8').decode('utf-8') # Eliminar caracteres especiales y números text = re.sub(r'[^\w\s]', '', text) text = re.sub(r'\d+', '', text) # Convertir a minúsculas text = text.lower() # Tokenizar, eliminar stopwords y lematizar stop\_words = set(stopwords.words('spanish')) tokens = [token.lemma\_ for token in nlp(text) if token.text not in stop\_words and token.is\_alpha] return ' '.join(tokens) # 3. Aplicar NER con Flair def apply\_ner(text): sentence = Sentence(text) tagger.predict(sentence) entities = [(entity.text, entity.labels[0].value) for entity in sentence.get\_spans('ner')] return entities # 4. Contar frecuencia de entidades def count\_entities(entities): counter = Counter(entities) data = [{'Entidad': ent, 'Tipo': typ, 'Frecuencia': freq} for (ent, typ), freq in counter.items()] df = pd.DataFrame(data) return df # 5. Guardar resultados en Excel def save\_to\_excel(df, output\_path='/content/NER\_Frecuencias.xlsx'): df.to\_excel(output\_path, index=False) print(f'Resultados guardados en: {output\_path}') # Pipeline completo text = read\_pdf(pdf\_path) clean\_text = preprocess\_text(text) entities = apply\_ner(clean\_text) result\_df = count\_entities(entities) print(result\_df) save\_to\_excel(result\_df). Qué hace este código?