Business Contract Validation

Aim : To classify content within the Contract Clauses & to determine deviations from Template & highlight them.

**Overview**

This script is designed to classify the content and individual clauses of a contract using Natural Language Processing (NLP). The script extracts text from a PDF contract, uses a pre-trained NLP model for classification, and detects deviations from a template contract.

**Requirements**

Ensure you have the following Python libraries installed:

* PyPDF2
* transformers
* torch
* PyMuPDF

You can install these libraries using pip:

|  |
| --- |
| pip install PyPDF2 transformers torch PyMuPDF |

**Script Sections**

1. **Import Libraries**

|  |
| --- |
| import PyPDF2  from transformers import pipeline  import fitz # PyMuPDF |

* **PyPDF2**: For reading PDF files.
* **transformers**: For loading the NLP model.
* **fitz**: (PyMuPDF) for additional PDF processing if needed.

1. **Extract Text from PDF**

|  |
| --- |
| # Load the PDF file  pdf\_file\_path = 'BusinessContract/BusinessContract2.pdf'  try:  # Open the PDF file in binary mode  with open(pdf\_file\_path, 'rb') as pdf\_file:  # Initialize the PDF reader  pdf\_reader = PyPDF2.PdfReader(pdf\_file)  # Initialize a variable to hold the extracted text  text = ""  # Iterate through all the pages and extract text  for page\_num in range(len(pdf\_reader.pages)):  page = pdf\_reader.pages[page\_num]  text += page.extract\_text()  # Print the extracted text  print(text)  except FileNotFoundError:  print(f"Error: The file '{pdf\_file\_path}' was not found.")  except PyPDF2.utils.PdfReadError as e:  print(f"Error reading PDF: {e}") |

* **pdf\_file\_path**: The path to the PDF file.
* The script opens the PDF, reads its content page by page, and extracts the text.

1. **Load NLP Model for Classification**

|  |
| --- |
| # Load a pre-trained NLP model for classification  classifier = pipeline("zero-shot-classification")  # Define candidate labels for content and clause classification  content\_labels = ["NDA", "Sales Agreement", "Employment Contract", "Lease Agreement"]  clause\_labels = ["Confidentiality", "Payment Terms", "Termination", "Liability"] |

* **classifier**: Uses a pre-trained model for zero-shot classification, which can classify text without specific training on the target categories.
* **content\_labels**: Possible classifications for the entire document.
* **clause\_labels**: Possible classifications for individual clauses.

**4. Classify Content of the Contract**

|  |
| --- |
| # Ensure text is not empty before classifying  if text.strip() and content\_labels:  # Classify the content of the contract  content\_result = classifier(text, content\_labels)  print("Content Classification:")  for label, score in zip(content\_result["labels"], content\_result["scores"]):  print(f"{label}: {score:.4f}")  else:  print("No valid text or labels to classify.") |

* The script checks if there is extracted text and valid labels.
* It classifies the entire content of the contract.

1. **Classify Individual Clauses**

|  |
| --- |
| # Split the contract into clauses (you'll need a more sophisticated method for real-world contracts)  clauses = text.split('\n\n')  # Classify each clause  for clause in clauses:  if clause.strip() and clause\_labels:  clause\_result = classifier(clause, clause\_labels)  print("\nClause Classification:")  for label, score in zip(clause\_result["labels"], clause\_result["scores"]):  print(f"{label}: {score:.4f}")  else:  print("Empty clause or no labels to classify.") |

* **clauses**: Splits the contract text into individual clauses based on double newlines.
* Each clause is classified using the clause labels.

1. **Deviation Detection**

|  |
| --- |
| # Deviation detection (requires a template contract for comparison)  # This is a simplified example, you'll need a more robust approach for real-world scenarios  template\_text = """  SOCIAL MEDIA MANAGEMENT CONTRACTUAL AGREEMENT  PARTIES  -This Social Media Management Contractual Agreement (hereinafter referred to as the "Agreement") is entered into on 12/12/9, by and between Digital Dreams, with an address of Andheri ,Mumbai (hereinafter referred to as the "Client") and John Smith, with an address of 123 Social Media Street, Suite 101, Social Media City, SM 12345, United States(hereinafter referred to as the "Social Media Manager") (collectively referred to as the "Parties").  SERVICES PROVIDED  The Social Media Manager agrees to provide comprehensive social media management services, tailored to the Client's specific needs and objectives. These services include but are not limited to:  1. Development of a customized social media strategy aligned with the Client's brand identity and marketing goals.  2. Creation of engaging and relevant content for distribution across various social media platforms, including but not limited to Facebook, Instagram, Twitter, LinkedIn, and TikTok.  3. Regular posting and scheduling of content to maximize audience engagement and reach.  4. Monitoring and moderation of social media channels to ensure timely responses to comments, messages, and inquiries.  5. Implementation of targeted advertising campaigns to increase brand visibility, drive website traffic,  """  template\_clauses = template\_text.split('\n\n')  for i, (clause, template\_clause) in enumerate(zip(clauses, template\_clauses)):  if clause != template\_clause:  print(f"\nDeviation detected in clause {i+1}") |

* **template\_text**: The text of a template contract.
* **template\_clauses**: Splits the template contract text into clauses.
* The script compares each clause from the contract with the corresponding clause from the template to detect deviations.

Submitted by:

* Vedant Dange(AI)
* Srushti Yawale(AI)
* Sokel Khan(AI)
* Anshul Suryawanshi(AI)