This script is designed to extract text from PDF and DOCX files, performing Optical Character Recognition (OCR) on scanned PDFs if necessary. The extracted text is then saved in a machine-readable JSON format. The script uses the following libraries:

- `easyocr`: For performing OCR on images.
- `pdfplumber`: For extracting text from PDF files.
- `docx2txt`: For extracting text from DOCX files.
- `numpy`: For handling image data in array format.
- `PIL` (Python Imaging Library): For image processing.
- `colorama`: For colored terminal output.

Code Breakdown

Imports and Initialization

```python
import easyocr
import pdfplumber
import docx2txt
import os
import json
import numpy as np
from PIL import Image
from colorama import Fore, Style, init

# Initialize colorama for colored output
init(autoreset=True)

# Create an EasyOCR Reader instance
reader = easyocr.Reader(['en'])  # Specify languages (e.g., 'en' for English)
```

- The required libraries are imported.
- `colorama` is initialized for colored terminal outputs to make log messages more readable.
- An instance of the EasyOCR reader is created, specifying English as the language for text recognition.

Function: `is_machine_readable`

```python
def is_machine_readable(filepath):
    """ Check if the document is already machine-readable by its extension. """
    _, ext = os.path.splitext(filepath)
    return ext.lower() in ['.txt', '.json']
```

- **Purpose**: Determine if the document is already in a machine-readable format based on its file extension.
- **Parameters**:
    - `filepath`: The path to the document file.

- **Returns**: `True` if the file is a `.txt` or `.json`, `False` otherwise.

Function: `extract_text_from_pdf`

```python
def extract_text_from_pdf(filepath):
    """ Extract text from a PDF document using pdfplumber. """
    text = ""
    print(f"{Fore.BLUE}Extracting text from PDF: {filepath}")

    with pdfplumber.open(filepath) as pdf:
        for i, page in enumerate(pdf.pages):
            page_text = page.extract_text()
            if page_text:
                print(f"{Fore.GREEN}Extracted text from page {i + 1} using pdfplumber.")
                text += page_text + "\n"
            else:
                print(f"{Fore.YELLOW}No text found on page {i + 1}. Performing OCR...")
                # If no text is found, apply OCR to the page image
                image = page.to_image(resolution=300)
                ocr_text = ocr_image(image.original)
                text += ocr_text + "\n"

    return text
```

- **Purpose**: Extract text from a PDF file. If a page does not contain text, it uses OCR to extract text from the page image.
- **Parameters**:
  - `filepath`: The path to the PDF file.
- **Returns**: A string containing all the extracted text from the PDF.

Function: `ocr_image`

```python
def ocr_image(image):
    """ Perform OCR on a given image using EasyOCR. """
    # Convert the image to a NumPy array for EasyOCR
    image_np = np.array(image)
    result = reader.readtext(image_np)
    return " ".join([res[1] for res in result])
```

- **Purpose**: Perform OCR on a given image to extract text.
- **Parameters**:
  - `image`: A PIL Image object.
- **Returns**: A string containing the extracted text from the image.

Function: `extract_text_from_docx`

```python
def extract_text_from_docx(filepath):
    """ Extract all text content from a DOCX file. """
    print(f"{Fore.YELLOW}Extracting text from DOCX: {filepath}")
    text = docx2txt.process(filepath)
    print(f"{Fore.GREEN}Text extraction from DOCX complete.")
    return text
```

- **Purpose**: Extract text from a DOCX file.
- **Parameters**:
  - `filepath`: The path to the DOCX file.
- **Returns**: A string containing all the extracted text from the DOCX file.

Function: `transform_to_machine_readable`

```
def transform_to_machine_readable(filepath):
    """ Transforms non-machine-readable documents into a JSON format. """
    _, ext = os.path.splitext(filepath)
    print(f"{Fore.BLUE}Starting transformation of {filepath} to machine-readable format...")

    if ext.lower() == '.pdf':
        text = extract_text_from_pdf(filepath)
    elif ext.lower() == '.docx':
        text = extract_text_from_docx(filepath)
    else:
        raise ValueError(f"{Fore.RED}Unsupported file type for OCR: {ext}")

    # Structure extracted text into JSON
    data = {"content": text}
    json_output = filepath.replace(ext, ".json")
    with open(json_output, 'w') as f:
        json.dump(data, f)

    print(f"{Fore.CYAN}Transformed and saved as JSON: {json_output}")
    return json_output
```

- **Purpose**: Convert non-machine-readable documents (PDFs or DOCX) into a machine-readable JSON format.
- **Parameters**:
  - `filepath`: The path to the document file.
- **Returns**: The path of the created JSON file.

Function: `process_document`

```
def process_document(filepath):
    """ Primary function to process documents based on their readability status. """
    print(f"{Fore.MAGENTA}Processing document: {filepath}")

    if is_machine_readable(filepath):
        print(f"{Fore.GREEN}{filepath} is already machine-readable.")
    elif os.path.splitext(filepath)[1].lower() in ['.pdf', '.docx']:
        print(f"{Fore.RED}{filepath} is not machine-readable. Transforming...")
        transform_to_machine_readable(filepath)
    else:
        print(f"{Fore.RED}Unsupported file type: {filepath}")
```

- **Purpose**: Main function to determine whether a document needs processing or if it is already in a machine-readable format.
- **Parameters**:
  - `filepath`: The path to the document file.
- **Functionality**: Calls other functions to handle the document based on its current format.

Example Usage

```
# Example usage - replace with actual document path
file_path = r"document_87ab3de3-dd28-476f-aab2-5967245f8e88 (2)-6-8.pdf"  # Update with you
process_document(file_path)
```

- This part of the code demonstrates how to use the `process_document` function with a specified file path. Update the `file_path` variable to point to your desired PDF or DOCX document.

Conclusion

This script automates the extraction of text from PDF and DOCX files, converting them into a machine-readable format. It is particularly useful for dealing with scanned documents where OCR is needed. Beginners can build upon this foundation by adding more features or improving the error handling as they become more familiar with Python and these libraries.