Configuration and Setup Guide for unstructured Local Inference Environment

I. Introduction

This document provides the necessary configuration artifacts and detailed, step-by-step instructions for establishing a reproducible conda environment tailored for utilizing the unstructured Python library with its local inference capabilities (unstructured[local-inference]) on a macOS system using Homebrew for system dependencies. The guide addresses specific version constraints and installation procedures derived from known compatibility requirements, particularly concerning NumPy 2.x and the Detectron2 library, ensuring a stable and functional setup for document processing tasks.

II. requirements.txt File Content

The following content should be placed in a file named requirements.txt. This file specifies the necessary Python packages and their version constraints for the environment. Adhering to these specifications is crucial for reproducibility and avoiding known compatibility issues.

Core unstructured library with local inference dependencies

Using quotes is essential for correct parsing of extras by pip

unstructured[local-inference]

Jupyter kernel support

ipykernel

NumPy: *Pinned below 2.0 due to widespread C-API/ABI incompatibility issues*

introduced in NumPy 2.0, affecting compiled dependencies like onnxruntime, torch, etc., leading to errors like '_ARRAY_API not found'.

This pin ensures stability within the specified package ecosystem.

numpy<2.0

Charset-Normalizer: *Pinned below 3.0 as requested.*

This prevents potential conflicts and ensures compatibility with specific

dependency versions within the unstructured stack.

charset-normalizer<3.0

Tesseract OCR wrapper for Python

pytesseract

Detectron2: *Required for certain layout models in unstructured[local-inference].*

Must be installed from this specific commit hash via Git for compatibility,

as official releases may have issues or different dependencies.

git+https://github.com/facebookresearch/detectron2.git@57bdb21249d5418c130d54e2ebdc94dda7a4c01a

This requirements.txt file serves not only as a build artifact but also as essential documentation. The included comments clarify the rationale behind specific version pins (like numpy<2.0) and non-standard installation sources (like the detectron2 git URL). This context is vital for maintainability and for understanding the environment's configuration, especially when troubleshooting or considering future updates. The NumPy constraint below version 2.0 is particularly important due to significant C-API and ABI changes introduced in NumPy 2.0, which break compatibility with many widely used compiled libraries, including onnxruntime (a likely dependency for local inference) and potentially torch (required by Detectron2). Downgrading NumPy (numpy<2) is the recommended solution when encountering related errors like AttributeError: _ARRAY_API not found. Installing unstructured[local-inference] requires quotes around the package name to ensure pip correctly interprets the extra dependencies specified within the brackets. detectron2 installation from a specific git commit hash is mandated by unstructured-inference documentation, likely due to the

complexity of its dependencies and potential lack of compatible official releases.

III. Step-by-Step Setup Guide

Follow these steps meticulously to create and configure the unstructured_env conda environment on a macOS system equipped with Homebrew.

A. Conda Environment Creation and Activation

Creating an isolated conda environment is the foundational step. This practice prevents conflicts between project dependencies and ensures that the specific versions required for unstructured[local-inference] do not interfere with other Python projects or the system's base Python installation. It is particularly crucial here due to the strict version pinning needed for NumPy.

1. **Create the conda environment:** Open your terminal and execute the following command. Using a specific, stable Python version like 3.10 is often recommended for compatibility with complex machine learning libraries.
   Bash
   `conda create -n unstructured_env python=3.10 -y`

2. **Activate the environment:** Before installing any packages, activate the newly created environment. All subsequent pip install commands in this guide must be run within this activated environment to ensure packages are installed correctly.
   Bash
   `conda activate unstructured_env`
   Your terminal prompt should now indicate that the unstructured_env is active.

B. System Dependency Installation (Homebrew)

The unstructured library relies on external, system-level tools for certain functionalities, especially for processing PDFs and images. These must be installed separately using a system package manager like Homebrew on macOS.

1. **Install dependencies:** Run the following command in your terminal:
   Bash
   `brew install poppler tesseract libmagic`

2. **Understand their roles:** These packages provide essential backend capabilities summarized below.

Dependency	Homebrew Package	Purpose in unstructured	Reference(s)

Poppler	poppler	Required for PDF processing, including rendering pages and extracting text or images (often utilized by the pdf2image Python library).	
Tesseract OCR	tesseract	Optical Character Recognition engine used to extract text from images or image-based PDFs (interfaced via pytesseract). Install tesseract-lang for additional language support.	
Libmagic	libmagic	Used for accurate detection of file types based on their content, improving the reliability of automatic document partitioning.	

C. Python Package Installation (within unstructured_env)

With the unstructured_env activated and system dependencies installed, proceed to install the required Python packages using pip. The order presented below is recommended to manage dependencies effectively, particularly the strict NumPy version constraint.

1. **Install NumPy (Pinned):** Install NumPy first, ensuring it remains below version 2.0 to avoid the aforementioned compatibility issues.
   Bash
   `pip install "numpy<2.0"`

2. **Install Charset-Normalizer (Pinned):** Install the user-requested pinned version.
   Bash
   `pip install "charset-normalizer<3.0"`

3. **Install Unstructured with Local Inference:** Install the core library and its local inference extras. Remember to use quotes. This step will pull in various dependencies like unstructured-inference, potentially onnxruntime, pdf2image, etc.
   Bash
   pip install "unstructured[local-inference]"

4. **Install Detectron2 from Git:** Install detectron2 from the specific commit hash required for compatibility. This requires build tools (like a C++ compiler) to be available on your system.
   Bash
   pip install "git+https://github.com/facebookresearch/detectron2.git@57bdb21249d5418c130d54e2ebdc94dda7a4c01a"

5. **Install ipykernel:** Required for integrating this environment with Jupyter Notebook/Lab.
   Bash
   pip install ipykernel

6. **Install Pytesseract:** The Python wrapper for the Tesseract OCR engine.
   Bash
   pip install pytesseract

The following table summarizes the key Python packages installed:

Package	Version/Source	Purpose	Key Rationale/Reference
numpy	<2.0	Fundamental numerical operations.	Pinned to avoid NumPy 2.x C-API/ABI incompatibility affecting compiled dependencies (e.g., onnxruntime, torch).
charset-normalizer	<3.0	Character set detection library.	Pinned as requested by user; prevents potential future conflicts.

unstructured[local-inference]	Latest compatible	Core library + dependencies for local model inference (layout, OCR).	Quotes required for pip extras syntax. Provides local processing capabilities.
detectron2	git+... @57bdb21...	Object detection library used for some layout models.	Installation from specific git commit required for compatibility with unstructured-inference.
ipykernel	Latest compatible	Enables environment use as a Jupyter kernel.	Standard package for Jupyter integration.
pytesseract	Latest compatible	Python wrapper for the Tesseract OCR engine.	Allows Python code to invoke the system-installed Tesseract for OCR tasks. Note: unstructured may use its fork if issues arise.

D. System PATH Verification

For unstructured (and its dependencies like pdf2image) to successfully call external tools like those provided by Poppler, the directory containing these executables must be listed in the system's PATH environment variable. Homebrew typically installs executables to /opt/homebrew/bin (Apple Silicon) or /usr/local/bin (Intel Macs).

1. **Check PATH:** Verify if the Homebrew binary directory is in your PATH.
   Bash
   ```
   echo $PATH
   ```
   Look for /opt/homebrew/bin or /usr/local/bin in the output.
2. **Update PATH (if necessary):** If the directory is missing, add it to your shell configuration file (e.g., ~/.zshrc for Zsh, ~/.bash_profile or ~/.bashrc for Bash). Add a line like the following (adjust path if needed):
   Bash
   ```
   export PATH="/opt/homebrew/bin:$PATH"
   ```
3. **Apply Changes:** Save the file and restart your terminal, or source the

configuration file (e.g., source ~/.zshrc). Any running JupyterLab instances should also be restarted for the new PATH to take effect in their processes. Failure to ensure the correct PATH can lead to errors when processing PDFs, even if Poppler is installed via Homebrew.

E. JupyterLab Kernel Integration

To use the unstructured_env from within a JupyterLab session (often launched from your base conda environment or another dedicated environment), you need to make JupyterLab aware of this new kernel. This involves installing packages in both the environment running JupyterLab and the target kernel environment (unstructured_env).

1. **Install nb_conda_kernels:** Activate the environment you use to launch JupyterLab (e.g., base) and install nb_conda_kernels. This package allows JupyterLab to automatically detect other conda environments that have ipykernel installed.
   Bash
   ```
   # Example assuming JupyterLab is launched from 'base'
   conda activate base
   conda install nb_conda_kernels
   # Reactivate your working environment if needed
   conda activate unstructured_env
   ```
2. **Confirm ipykernel:** Ensure ipykernel is installed in the target environment (unstructured_env). This was completed in step III.C.5. ipykernel is what makes the environment capable of acting as a Jupyter kernel.
3. **Restart JupyterLab:** If JupyterLab was running during the installation of nb_conda_kernels or the creation/modification of unstructured_env, restart it completely. This allows the nb_conda_kernels extension to scan for and register the new environment.
4. **Verify:** After restarting, launch JupyterLab. You should now see an option like "Python [conda env:unstructured_env]" or similar in the Launcher window when creating a new notebook, or in the "Kernel" > "Change Kernel..." menu.

IV. Verification (Recommended)

After completing the setup, it is advisable to verify that the core components are installed correctly and accessible within the environment.

1. **Launch JupyterLab** (if not already running).
2. **Create a new Notebook** using the "Python [conda env:unstructured_env]"

kernel.

3. **Execute the following code** in a notebook cell:

```python
import numpy
import unstructured
from unstructured.partition.auto import partition
import os
import sys

print(f"--- Environment Verification ---")
print(f"Python Executable: {sys.executable}")
print(f"NumPy version: {numpy.__version__}")
print(f"Unstructured library imported successfully.")

# Check if detectron2 related components can be imported (optional but good check)
try:
    # Attempt to import something dependent on detectron2 via unstructured_inference
    from unstructured_inference.models.detectron2 import
UnstructuredDetectronModel
    print("Detectron2-related import successful.")
except ImportError as e:
    print(f"Detectron2-related import failed: {e}")
    print("(This might be acceptable if not directly using Detectron2-based models).")
except Exception as e:
    print(f"An unexpected error occurred during Detectron2 import check: {e}")

# Simple check for PATH accessibility of poppler (indirectly via pdf2image)
try:
    from pdf2image.exceptions import PDFInfoNotInstalledError
    from pdf2image import pdfinfo_from_bytes
    # Create a dummy PDF byte stream (minimal valid PDF)
    dummy_pdf_bytes = b'%PDF-1.0\n1 0 obj<</Type/Catalog/Pages 2 0 R>>endobj 2 0
obj<</Type/Pages/Count 0>>endobj\nxref\n0 3\n0000000000 65535 f\n0000000010 00000
n\n0000000058 00000 n\ntrailer<</Size 3/Root 1 0 R>>\nstartxref\n106\n%%EOF'
    info = pdfinfo_from_bytes(dummy_pdf_bytes, userpw=None,
poppler_path=None)
    print("Poppler tools seem accessible via PATH.")
except PDFInfoNotInstalledError:
    print("ERROR: Poppler tools (pdfinfo) not found in PATH. PDF processing will likely fail.")
    print("Ensure Homebrew bin directory is in your $PATH and JupyterLab was restarted.")
```

```
except Exception as e:
    print(f"An unexpected error occurred during Poppler check: {e}")

print(f"--- Verification Complete ---")
```

4. **Interpret Results:** Successful execution without ImportError for numpy and unstructured, confirmation of numpy version <2.0, and ideally a successful Detectron2-related import and Poppler tools seem accessible message indicate a correctly configured environment. Errors during the import checks or the Poppler check point towards issues with package installation or the system PATH. This verification step provides immediate feedback on the complex installation process.

V. Conclusion

Following the steps outlined in this guide should result in a functional and reproducible conda environment named unstructured_env on macOS. This environment is specifically configured to run unstructured[local-inference] while addressing known compatibility challenges, particularly the pinning of numpy<2.0 and the specific installation method for detectron2. Utilizing the provided requirements.txt file facilitates recreating this environment consistently.

While this guide focuses on macOS with Homebrew, setting up a similar environment on Linux or Windows would require adjustments, primarily in the installation commands for system dependencies (using apt, yum, or potentially manual installations) and potentially facing greater challenges with detectron2 installation on Windows. The core Python package requirements and version constraints, especially for NumPy, are likely to remain relevant across platforms.