

The error message `RuntimeError: Error running shell script install_openmm.sh`, output in log file `install_log.txt` indicates that a shell script, specifically `install_openmm.sh`, failed to execute successfully within the `run_shell_commands` function. The script is responsible for installing OpenMM for AMBER refinement using Conda.

Here's a step-by-step breakdown of the potential causes:

1. **Conda Environment:** The script attempts to install OpenMM within a Conda environment. If the Conda environment is not properly set up or activated, the installation might fail. This could be due to issues with Conda's installation, configuration, or previous errors that have corrupted the environment.
2. **Dependency Conflicts:** OpenMM has dependencies on other packages, and there might be conflicts with existing packages in the Conda environment. These conflicts can prevent the installation or lead to errors during runtime.
3. **Network Issues:** The script downloads packages from the internet, and network connectivity problems, such as firewalls or proxy settings, can disrupt the process.
4. **Insufficient Permissions:** The script might require certain permissions to execute commands or modify files on the system. If these permissions are not granted, the installation can fail.
5. **Incorrect Python Version:** The script attempts to install OpenMM for a specific Python version. There might be an issue if the Colab environment has a different Python version installed or if the script incorrectly identifies the Python version.

Suggested Changes

`content_copy`

```
def run_shell_commands(commands, filename, install_log):  
    with open(filename, 'w') as f:  
        f.write(commands)  
  
    # Use the get_ipython().system_raw method instead of get_ipython().system  
    # This allows us to capture the exit code and check for errors.  
    import subprocess
```

```
result = subprocess.run(['bash', '-x', filename], stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)
```

```
# Write the output and error messages to the log file.
```

```
with open(install_log, 'a') as f:
```

```
    f.write(result.stdout)
```

```
    f.write(result.stderr)
```

```
if result.returncode != 0:
```

```
    raise RuntimeError('Error running shell script %s, output in log file %s'
```

```
                      % (filename, install_log))
```

[Use code with caution](#)

Explanation of changes:

1. **Error Handling:** I have replaced `get_ipython().system` with `subprocess.run` to execute the shell script and capture the exit code. This allows us to check for errors during the script's execution and raise a `RuntimeError` if the exit code is not 0.
2. **Detailed Logging:** The `subprocess.run` captures the stdout and stderr and writes them to the install log file for easier debugging.
3. **Conda Environment:** Make sure that the script uses the correct Conda environment and activates it before installing OpenMM. If necessary, create a new Conda environment for this task to avoid conflicts with other packages. Consider explicitly specifying the channel for OpenMM during installation, which can help resolve dependency issues. For example:

content_copy

```
conda install -y -q -c conda-forge openmm=7.7.0 python={python_version} pdbfixer 2>&1 1>/dev/null
```

[Use code with caution](#)

By implementing these changes, you improve the error handling and logging capabilities of the `run_shell_commands` function, increasing the chances of identifying and resolving the root cause of the `RuntimeError`.

Rate this answer [thumb_up](#) [thumb_down](#)

