

# Appendix for Beyond pip install: Evaluating LLM agents for the automated installation of Python projects

In this document, we provide supplementary materials to the main content of the paper. This includes the prompts used by Installamatic, and three tables: the first provides a brief example of the type of commands in the exemplar Dockerfiles that would warrant assigning each tag; the second table shows detailed results for the first experiment conducted, in which Installamatic is tasked with installing the repositories in the dataset after performing the document gathering step; Finally, the third table shows the results for the second experiment, in which the document gathering step is skipped, and all install-relevant documents are provided to the Installamatic automatically. The second experiment excludes the `icloud-drive-docker`, `instructor`, `yfinance`, `tqdm`, `core` and `sherlock` repositories. This is due to the fact that these repositories do not contain any install-relevant information, and therefore the perfect recall scenario would not apply to them.

## I. DESCRIPTION AND EXAMPLE OF EACH TAG

| Tag                 | Description  | Example  |
|---------------------|--|--|
| <b>Installation</b> |  |  |
| requirements        | Installation of dependencies using <code>pip install -r requirements.txt</code> .  | RUN <code>pip install -r requirements.txt</code>   |
| requirements-extra  | Installation of dependencies from additional requirements.   | RUN <code>pip install -r test_requirements.txt</code>  |
| pip-extra           | Requiring the use of pip to install of something other than Poetry or the contents of a requirements file.   | RUN <code>pip install requests_cache requests_ratelimiter</code>   |
| poetry              | Installation of dependencies using the Poetry dependency manger  | RUN <code>pip install poetry</code><br>RUN <code>poetry install</code><br>RUN <code>poetry run ...</code>                        |
| poetry-extra        | Installation of dependencies using Poetry, with additional arguments.  | RUN <code>pip install poetry</code><br>RUN <code>poetry install --with dev,docs -E all</code><br>RUN <code>poetry run ...</code> |
| make-install        | Installation of dependencies using a makefile, typically commands such as <code>make install</code> or <code>make init</code> .                                    | RUN <code>make init</code>   |
| install-self        | Achieved by running <code>pip install -e .</code> , this means that the project itself needs to be installed in the working environment in order for tests to run. | RUN <code>pip install --no-build-isolation --editable .</code>   |
| install-pytest      | The Pytest library needs to be installed manually.   | RUN <code>pip install pytest</code>  |
| install-tox         | The Tox library needs to be installed manually.  | RUN <code>pip install tox</code>   |
| install-other       | Perform installation of dependencies through other means, such as a custom script contained in the repository  | RUN <code>scripts/install</code>   |
| <b>Testing</b>      |  |  |
| pytest              | Tests are run using PyTest.  | RUN <code>pytest</code>  |
| pytest-extra        | Additional arguments need to be provided to pytest, such as specifying the location of the tests or additional flags.  | RUN <code>poetry run pytest --fast-test-mode.</code>   |
| tox                 | Tests are run using Tox.   | RUN <code>tox</code>   |
| unittest            | Tests are run using Python's built in unittest command.  | RUN <code>python -m unittest discover</code>   |
| make-test           | Tests are run using a makefile with a command such as <code>make test</code> .   | RUN <code>make test</code>   |
| test-other          | Tests are run some other way, such as a <code>test.py</code> file.   | RUN <code>python setup.py test</code>  |
| <b>Other</b>        |  |  |
| bash-extra          | Requiring additional bash commands to set up the repository, such as creating new directories or granting permissions to certain files.                            | ENV <code>OPENAI_API_KEY=x</code>  |

TABLE I: Description and example of each installation tag

## II. EXPERIMENT RESULTS

| Repository          | Build Rate | Average #Attempts | Average Duration (s) | Average Recall | Average #Relevant | Average #Irrelevant |
|---------------------|------------|-------------------|----------------------|----------------|-------------------|---------------------|
| mypy                | 9/10       | 1.6               | 898.821              | 0.9            | 1                 | 2.4                 |
| Torch-Pruning       | 0/10       | 2.5               | 405.783              | 1.0            | 1                 | 1.5                 |
| scapy               | 3/10       | 2.0               | 263.935              | 0.25           | 2                 | 1.4                 |
| ydata-profiling     | 2/10       | 2.6               | 676.187              | 0.45           | 2                 | 2.4                 |
| cloud-custodian     | 0/10       | 3.0               | 197.62               | 0.0            | 1                 | 1.0                 |
| black               | 4/10       | 2.4               | 317.092              | 0.0            | 1                 | 1.8                 |
| speechbrain         | 0/10       | 2.3               | 504.869              | 0.35           | 2                 | 2.6                 |
| camel               | 0/10       | 2.5               | 250.867              | 0.5            | 3                 | 0.1                 |
| open-interpreter    | 8/10       | 1.9               | 200.01               | 0.7            | 1                 | 0.6                 |
| sabnzbd             | 0/10       | 2.5               | 297.514              | 0.9            | 1                 | 1.1                 |
| sherlock            | 1/10       | 2.6               | 113.326              | 0.0            | 0                 | 1.0                 |
| pymc                | 0/10       | 2.8               | 808.95               | 0.0            | 2                 | 2.7                 |
| pennylane           | 0/10       | 2.4               | 246.368              | 0.033          | 3                 | 2.2                 |
| beets               | 7/8        | 2.25              | 129.139              | 0.875          | 1                 | 0.5                 |
| instructor          | 0/10       | 2.9               | 235.4                | 0.0            | 0                 | 2.2                 |
| scvi-tools          | 0/10       | 2.5               | 912.604              | 0.3            | 1                 | 1.7                 |
| boto3               | 0/10       | 3.0               | 1090.803             | 0.8            | 1                 | 1.3                 |
| tqdm                | 4/10       | 2.3               | 254.542              | 0.0            | 0                 | 2.2                 |
| moto                | 6/10       | 1.5               | 1470.263             | 0.233          | 3                 | 2.3                 |
| X-AnyLabeling       | 2/10       | 2.4               | 282.206              | 0.2            | 1                 | 2.5                 |
| fastapi             | 7/10       | 1.9               | 150.565              | 0.0            | 2                 | 2.3                 |
| sympy               | 0/10       | 2.6               | 3009.718             | 0.5            | 2                 | 1.0                 |
| yfinance            | 0/10       | 2.8               | 139.954              | 0.0            | 0                 | 2.1                 |
| R2R                 | 0/10       | 1.7               | 210.39               | 0.2            | 1                 | 1.5                 |
| rich                | 7/10       | 2.3               | 118.019              | 1.0            | 1                 | 0.1                 |
| numba               | 0/10       | 2.5               | 145.67               | 0.0            | 2                 | 2.1                 |
| dlt                 | 0/10       | 2.9               | 452.216              | 1.0            | 1                 | 0.5                 |
| aim                 | 0/10       | 2.3               | 348.748              | 1.0            | 1                 | 0.9                 |
| qlib                | 0/10       | 2.5               | 775.61               | 0.5            | 2                 | 0.9                 |
| textual             | 10/10      | 1.7               | 387.686              | 1.0            | 1                 | 0.0                 |
| nonebot2            | 0/10       | 3.0               | 249.332              | 0.2            | 1                 | 1.0                 |
| opencompass         | 1/10       | 2.7               | 568.671              | 0.5            | 2                 | 0.5                 |
| django-stubs        | 8/10       | 2.1               | 291.474              | 1.0            | 1                 | 1.1                 |
| you-get             | 8/10       | 2.1               | 106.523              | 0.8            | 1                 | 1.6                 |
| spotify-downloader  | 9/10       | 1.6               | 329.153              | 0.5            | 2                 | 1.8                 |
| core                | 0/10       | 2.0               | 365.041              | 0.0            | 0                 | 2.3                 |
| starlette           | 8/10       | 1.7               | 98.334               | 0.4            | 2                 | 2.0                 |
| datasets            | 0/10       | 2.7               | 837.882              | 0.5            | 1                 | 0.7                 |
| spaCy               | 6/10       | 2.2               | 610.086              | 1.0            | 1                 | 1.4                 |
| icloud-drive-docker | 0/10       | 2.1               | 127.176              | 0.0            | 0                 | 2.8                 |

TABLE II: Full table of results

| Repository         | Build Rate | Average # Attempts | Average Duration (s) | Average #Relevant |
|--------------------|------------|--------------------|----------------------|-------------------|
| qlib               | 0/10       | 3.0                | 619.722              | 2                 |
| cloud-custodian    | 0/10       | 2.8                | 176.226              | 1                 |
| fastapi            | 10/10      | 1.0                | 132.549              | 2                 |
| nonebot2           | 0/10       | 3.0                | 109.311              | 1                 |
| sabnzbd            | 0/10       | 2.6                | 416.891              | 1                 |
| spotify-downloader | 9/10       | 1.3                | 314.772              | 2                 |
| sympy              | 0/10       | 2.6                | 2869.016             | 2                 |
| pymc               | 0/10       | 2.4                | 238.755              | 2                 |
| rich               | 9/10       | 1.9                | 90.82                | 1                 |
| mypy               | 8/10       | 2.5                | 805.807              | 1                 |
| scapy              | 9/10       | 1.3                | 213.261              | 2                 |
| aim                | 2/10       | 2.8                | 388.659              | 1                 |
| django-stubs       | 8/10       | 2.3                | 150.823              | 1                 |
| ydata-profiling    | 1/10       | 2.8                | 577.774              | 2                 |
| boto3              | 2/10       | 2.5                | 398.713              | 1                 |
| textual            | 6/10       | 2.5                | 276.669              | 1                 |
| camel              | 0/10       | 2.8                | 520.734              | 3                 |
| numba              | 0/10       | 2.8                | 878.6                | 2                 |
| black              | 3/10       | 2.1                | 466.913              | 1                 |
| open-interpreter   | 10/10      | 1.1                | 74.927               | 1                 |
| datasets           | 0/10       | 3.0                | 1872.915             | 1                 |
| opencompass        | 0/10       | 2.6                | 372.999              | 2                 |
| scvi-tools         | 0/10       | 2.4                | 1548.367             | 1                 |
| dlt                | 0/10       | 2.8                | 325.814              | 1                 |
| moto               | 9/10       | 1.5                | 1211.358             | 3                 |
| you-get            | 7/10       | 2.2                | 185.332              | 1                 |
| starlette          | 10/10      | 1.0                | 72.231               | 2                 |
| pennylane          | 0/10       | 2.6                | 428.72               | 3                 |
| spaCy              | 8/10       | 2.4                | 970.298              | 1                 |
| speechbrain        | 0/10       | 2.8                | 493.776              | 2                 |
| X-AnyLabeling      | 4/10       | 2.2                | 413.028              | 1                 |
| beets              | 3/10       | 2.9                | 136.428              | 1                 |
| R2R                | 0/10       | 2.4                | 109.008              | 1                 |
| Torch-Pruning      | 0/10       | 3.0                | 440.809              | 1                 |

TABLE III: Full table of results for run with search step skipped (perfect recall)

### III. PROMPTS

For the sake of clarity, the prompts have been split into three groups: the documentation gathering step, Dockerfile generation and Dockerfile repair.

#### A. Documentation Gathering

I want to write a dockerfile to build the <REPO\_NAME> project.  
Your task is to search the given repository using the provided tools and collect all files that contain documentation related to environment setup, installing dependencies and running tests.  
You must try to install the repository using pip. 'pip install <REPO\_NAME>' is NOT what we want to find, this is INCORRECT and IRRELEVANT to building the project from source, which is our goal.  
Such content could be found in files with names like "testing", "contributing", "setup", etc. Collecting an irrelevant file could harm future results, so make sure that any file you register is indeed relevant.  
To reiterate, a file that makes explicit mention of **how to install the project's dependencies** or how to **run unit tests** is considered relevant. Anything else should be ignored.  
Your focus should be on natural language documents. DO NOT attempt to read or register python files, for example.  
Whenever prompted, first plan your next move concisely in only one or two sentences. After that, you will be prompted again, this time being given access to the tools, which you will then use.  
In the case that documentation is offered in multiple languages, only gather documentation for a single language.

Here are the contents of the repository's root directory (`. `):  
<CONTENTS>

Fig. 1: Initial prompt for the documentation gathering step

In one or two sentences, give your thoughts about the provided information, then describe what you would like to do next.  
You can use provided tools to retrieve additional information about the repo's contents.  
Whenever you find a documentation file, you will submit it using the <SUBMIT\_TOOL> tool, then continue your search. The whole file is submitted at once, so even if a document contains multiple relevant sections, you only need to submit it once.  
Once you are confident that you have found all documentation files in the repository, use the <FINISHED\_SEARCH> tool to move on to your next task.

Fig. 2: Follow-up prompt for the documentation gathering step

## B. Dockerfile Generation

Believe it or not, the line about being eaten alive in the Dockerfile generation prompt (Figure 5) actually helped performance by providing emotional stimulus to the LLM<sup>1</sup>.

```
I want to write a docker file to place inside this repo that will set up a
development environment, install any dependencies and run tests to confirm it
works.
Remember, instructions such as 'pip install <REPO_NAME>' are NOT helpful. I want to
build the project from source.
Using the gathered files, collect and summarise any information that may help me. The
gathered files, which you now have access to are:
- <FILES>
You may use the <TOOLS> tools to reinspect the contents of these files if you wish,
but you can not access any files other than these. Once you are done, use the <
SUMMARISE_TOOL> to give a summary of the information you found.
```

Fig. 3: Initial prompt for documentation summarisation

```
In one or two sentences, give your thoughts about the provided information, then
describe what you would like to do next.
You can use provided tools to retrieve additional information about the repo's
contents.
You may use the <TOOLS> tools to reinspect the contents of these files if you wish,
but you can not access any files other than these. Once you are done, use the <
SUMMARISE_TOOL> to give a summary of the information you found.
```

Fig. 4: Follow-up prompt for documentation summarisation

```
Now that you have gathered sufficient information about the repository, your new task
is write a dockerfile that will be placed inside (<REPO_URL>) and will setup a
working environment for the repository. To verify that any cloning and
installation succeeds, 'RUN' the repo's test suite at the end of the build process
.
Here are some tips for successfully writing a dockerfile:
- Since the dockerfile will be placed inside the repository, there is no need to
clone the repo!
- To ensure that you dont miss any files important to installing the requirements,
make sure to copy the *entirety* of the repo into the container. ('COPY . /app/')
- make sure that the tests run during the build process, by using 'RUN' when running
tests. If the final line uses 'CMD' then I will be eaten alive (bad). DO NOT EVER
USE 'CMD <test command>'.
  - Dont add any additional arguments to the test command, that makes it harder for
me to identify whether the build was successful.
  - Make sure your final line is 'RUN <test command>'
use the 'submit_dockerfile' function to provide the dockerfile.
```

Fig. 5: Prompt for Dockerfile generation

<sup>1</sup><https://arxiv.org/abs/2307.11760>

### C. Dockerfile Repair

```
Attempting to build the project using this dockerfile is resulting in the following
error:
'''
<ERROR_LOG>
'''
In one or two sentences only, perform a concise diagnosis of this error. What do the
error logs mean, and what could be causing the error? You do not need to suggest a
fix to the dockerfile yet, just identify the error.
Some reminders that might be helpful:
<REPAIR_HINTS>
```

Fig. 6: System prompt for Dockerfile repair

```
- If you are testing with pytest, make sure the final line is 'RUN pytest'
- If you are using poetry, then 'RUN poetry run pytest'.
- If you are testing with make, the correct command is likely 'make test', not 'make
  tests'.
- trying to copy the requirements file from one place to another will often lead to
  mistakes. Avoid this too
- Make sure any files that are referenced really do exist. Here are the contents of
  the repo's root directory, '.' (not including the dockerfile, which you have
  already been shown):
<ROOT_DIRECTORY>
- It could be the case the repository has more than one requirements file, if (AND
  ONLY IF) you are certain the repository has multiple requirements files, it could
  be the case that the requirements from multiple files are needed to run tests.
- If you encounter an error like this: 'ERROR: Cannot find command 'git' - do you
  have 'git' installed and in your PATH?', then you need to install git like so: '
  apt install git'
- If you encounter an error that looks like the one shown below, it could be the case
  that you need to use a newer python version.
'''
#9 11.05 ERROR: Cannot install <DEPENDENCY>==<VER_NUM> because these package versions
  have conflicting dependencies.
#9 11.05
#9 11.05 The conflict is caused by:
#9 11.05     The user requested <DEPENDENCY>==<VER_NUM>
#9 11.05     The user requested (constraint) <DEPENDENCY>==<VER_NUM>
'''
- If you encounter an error such as No module named '<PROJECT_NAME>.xxx' , where the
  name of the missing module is also the name of the project you are setting up,
  then you may need to compile the project before testing.
- You can do this after installing the requirements like so: 'pip install wheel ; pip
  install --no-build-isolation --editable .'
```

Fig. 7: Additional hints to aid in repair

I am trying to build the following dockerfile for the <REPO\_URL> repository. It should install any necessary dependencies and then run tests to confirm that installation succeeded:  
<DOCKERFILE>

Fig. 8: Prompt for Dockerfile error diagnosis

Now, again in only one or two sentences, posit whether this could be fixed by adjusting the dockerfile, or if there is no way for you to build and test this repository.

Examples of errors that cannot be fixed:

- Tests requiring api keys that you do not have access to
- Errors in files in the repo, such as the requirements file.

After explaining the cause of the error, use the provided <READY\_TO\_FIX> tool to indicate that you are capable of fixing this error by adjusting the dockerfile. If you already have a fix in mind, use the provided <READY\_TO\_FIX> tool straight away to confirm that you believe the dockerfile is fixable, and proceed to the next step. Use the <READY\_TO\_FIX> tool the same way you would any other tool: provide a call to it when you are prompted to use a tool. Do not attempt to use the tool when you are still planning your next move in natural language. Otherwise (only if you **do not** already have an idea for a fix), you can use the other tools provided (<SEARCH\_TOOLS>) to inspect the contents of the repository for additional information. When using these tools, remember to give file paths **relative to the root directory of the project**, absolutely do not include `"/usr/src/app/<project_name>"` when providing file paths to the search tools.

Fig. 9: Prompt for search stage of Dockerfile error diagnosis

You can use the other tools provided (<SEARCH\_TOOLS>) to inspect the contents of the repository if you need any additional information. First plan your next action in one or two sentences, then you will be given access to the tools. After explaining the cause of the error, use the provided <READY\_TO\_FIX> tool to confirm that you are ready to suggest a fix to the dockerfile.

Fig. 10: Followup prompt for search stage of Dockerfile error diagnosis

Now, suggest how to fix this error, then use the provided tool to return a fixed version of the dockerfile.  
Recall the tips I gave you earlier:  
<REPAIR\_HINTS>

Fig. 11: Prompt for generating repaired Dockerfile