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 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
Installation		
requirements	Installation of dependencies using `pip install -r requirements.txt`.	`RUN pip install -r requirements.txt`
requirements-extra	Installation of dependencies from additional requirements.	`RUN pip install -r test_requirements.txt`
pip-extra	Requiring the use of pip to install of something other than Poetry or the contents of a requirements file.	`RUN pip install requests_cache requests_ratelimiter`

poetry	Installation of dependencies using the Poetry dependency manger	`RUN pip install poetry`
		`RUN poetry install`
		`RUN poetry run ...`

poetry-extra	Installation of dependencies using Poetry, with additional arguments.	`RUN pip install poetry`
		`RUN poetry install --with dev,docs -E all`
		`RUN poetry run ...`

make-install	Installation of dependencies using a makefile, typically commands such as `make install` or `make init`.	`RUN make init`
install-self	Achieved by running `pip install -e .`, this means that the project itself needs to installed in the working environment in order for tests to run.	`RUN pip install --no-build-isolation --editable .`
install-pytest	The Pytest library needs to be installed manually.	`RUN pip install pytest`
install-tox	The Tox library needs to be installed manually.	`RUN pip install tox`
install-other	Perform installation of dependencies through other means, such as a custom script contained in the repository	`RUN scripts/install`
Testing		
pytest	Tests are run using PyTest.	`RUN pytest`
pytest-extra	Additional arguments need to be provided to pytest, such as specifying the location of the tests or additional flags.	`RUN poetry run pytest --fast-test-mode.`
tox	Tests are run using Tox.	`RUN tox`
unittest	Tests are run using Python's built in unnittest command.	`RUN python -m unittest discover`
make-test	Tests are run using a makefile with a command such as `make test`.	`RUN make test`
test-other	Tests are run some other way, such as a `test.py` file.	`RUN python setup.py test`
Other		
bash-extra	Requiring additional bash commands to set up the repository, such as creating new directories or granting permissions to certain files.	ENV OPENAI_API_KEY=x

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)