



# Automated Extraction of Research Software Installation Instructions from README files

**Carlos Utrilla Guerrero**<sup>1,2</sup>, Oscar Corcho<sup>1</sup> and Daniel Garijo<sup>1</sup>

<sup>1</sup> Ontology Engineering Group, Universidad Politécnica de Madrid, Spain

<sup>2</sup> Research Data and Software Team, TU Delft University, The Netherlands



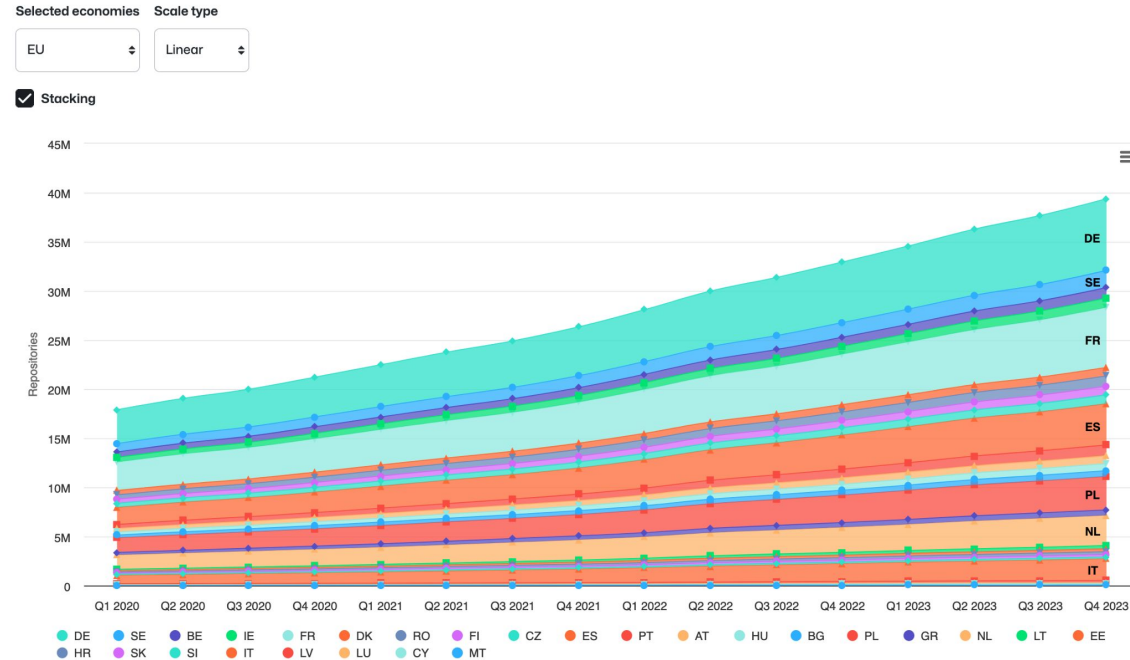
c.utrillaguerrero@tudelft.nl



<https://carlosug.github.io/>



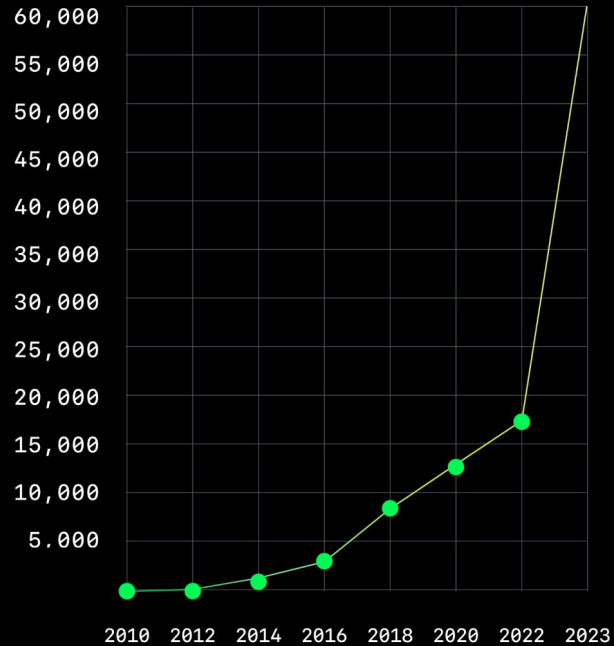
## Count of Repositories Among Economies in the EU [↗](#)



Every software repository comes with a **README file** (<https://docs.github.com/en/repositories>)

<https://innovationgraph.github.com/global-metrics/repositories>

## The global growth in generative AI projects on GitHub



GitHub is foundational to **how software is build** today

<https://github.blog/2023-11-08-the-state-of-open-source-and-ai/#the-explosive-growth-of-generative-ai-in-2023>

Percentage\* of repositories with and without README, OPEN SOURCE:

**85.9%**  
With README

**14.1%**  
No README

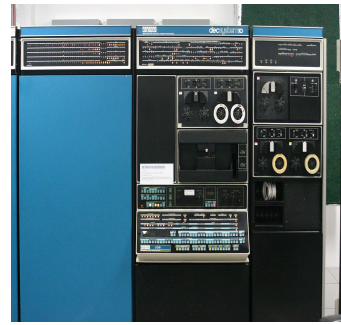
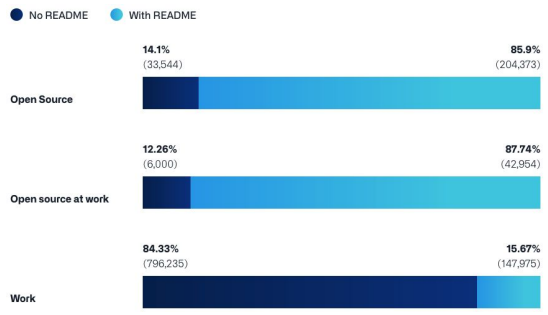
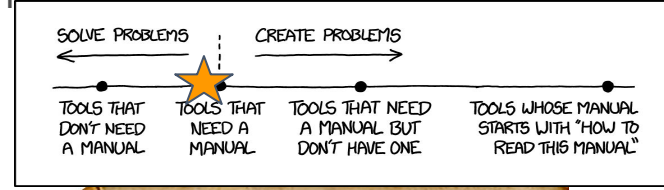
WRITING  
README



\* source: <https://octoverse.github.com/2021/creating-documentation/#readme-or-not>

# README: a file to facilitate research software understanding!

If we wish to investigate further, a clearly written, up-to-date documentation should be available for researchers to run interpretively



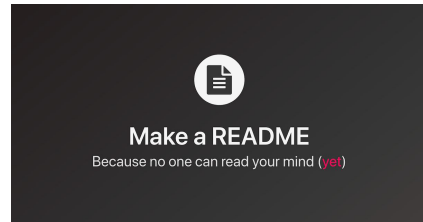
1974: PDP-10 software had "README.TXT"  
[http://pdp-10.trailing-edge.com/decus\\_20tap3\\_198111/01/decus/20-007\\_9/readme.txt.html](http://pdp-10.trailing-edge.com/decus_20tap3_198111/01/decus/20-007_9/readme.txt.html)

Sacrifice everything else to brevity: <https://xkcd.com/1343/>



README convention to the famous scene in Lewis Carroll's *Alice's Adventures In Wonderland* in which Alice confronts magic munchies labeled "Eat Me" and "Drink Me":  
<https://medium.com/@NSomar/readme-md-history-and-components-a365aff07f10>

<https://octoverse.github.com/2021/creating-documentation/#readme-or-not>



# Challenge: README is not foolproof, often is difficult to interpret!

- Instructions on how to install, setup, run and **test new ideas!**
- How to carry out the various individual operations involved in a installation process?

## Installation

Pykan can be installed via PyPI or directly from GitHub.

### Pre-requisites:

```
Python 3.9.7 or higher  
pip
```

### Installation via github

```
python -m venv pykan-env  
source pykan-env/bin/activate # On Windows use `pykan-env\Scripts\activate`  
pip install git+https://github.com/KindXiaoming/pykan.git
```

### Installation via PyPI:

```
python -m venv pykan-env  
source pykan-env/bin/activate # On Windows use `pykan-env\Scripts\activate`  
pip install pykan
```

### Requirements

```
# python==3.9.7  
matplotlib==3.6.2  
numpy==1.24.4  
scikit_learn==1.1.3  
setuptools==65.5.0  
sympy==1.11.1  
torch==2.2.2  
tqdm==4.66.2
```

Repo: <https://github.com/KindXiaoming/pykan>

After activating the virtual environment, you can install specific package requirements as follows:

```
pip install -r requirements.txt
```

**Optional: Conda Environment Setup** For those who prefer using Conda:

```
conda create --name pykan-env python=3.9.7  
conda activate pykan-env  
pip install git+https://github.com/KindXiaoming/pykan.git # For GitHub installation  
# or  
pip install pykan # For PyPI installation
```

# Challenge: READMEs may be difficult to interpret!

- Instructions on how to install, setup, run and test new ideas!
- How to carry out the various **individual operations** involved in a software installation process?

**Installation from source** → A type of installation method

Installation from source is the preferred way to set up SWE-agent on your machine.

!!! warning "Issues on Windows" Expect some issues with Windows (we're working on them). In the meantime, use Docker (see below).

1. Install Docker, then start Docker locally.
2. For the web interface only: Install `nodejs`.
3. Clone this repository.
4. Run `pip install --editable .` at the repository root (as with any python setup, it's recommended to use `conda` or `virtual environments` to manage dependencies).
5. Run `./setup.sh` to create the `swe-agent` docker image.
6. Create a `keys.cfg` file at the root of this repository ([more information](#)).

!!! tip "Docker issues" If you run into docker issues, see the [installation tips section](#) for more help

OK! Green Win users!

wasn't from source? requirements? which version nodejs?

3. execute in command line?
4. pip install -e? repository root, shall we config env before?
5. shall instructions be obeyed in the right order?
6. shall private API-keys be added on it?

source: <https://github.com/princeton-nlp/SWE-agent/blob/main/README.md>

Will change operations overtime?

**Challenges:** inconsistency, ambiguity, incompleteness, time-dependent

The art of **Guesstimate** the installation task!!

README file contain essential information written in **human language**, making automated extraction challenging

# Challenge: README is not foolproof, often is difficult to interpret!

- **Inconsistency in documentation practice:**
  - Diverse installation methods: pip, source, container, binary
  - Requirements and dependencies aren't part of installation procedures
  - Optional steps to configure environmental variables
- **Redundancy and/or incompleteness:**
  - Add useless and/or obsolete commands for file location
  - Ambiguity guessing about specific actions in root directory
  - Insufficient instructions for executing a command
- **Time-variant:**
  - Often instructions will begin to change/degrade in time or accuracy before it fails
  - The major time sink is removing package conflicts with version and configure environments
  - Methods stop working eventually if not actively maintained
  - Tools and libraries and dependencies changes may cause collapse
  - Bugs are found, new features are needed, new setup requirements arise

Critical information is missing for **effective software reuse**

**Authors tendency to oversimplify the role standards in documentation, making reuse of research software difficult**

**README file contain essential information written in human language, making automated extraction challenging**



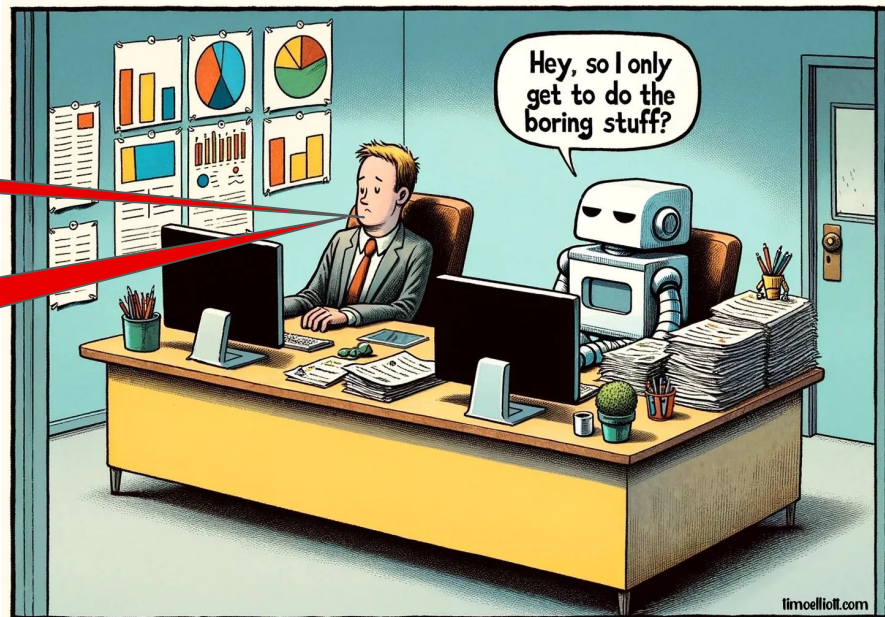
# Our goal: LLMs can serve as automated assistance in the research software setup?

LLMs to turn natural language instructions into a procedure that must be executed correctly. We aim to **understand** how well LLMs are able to:

- Detect alternate installation methods (**Plans**)
- Extract their sequential instructions (**Steps**)

YES  
PLEASE!!

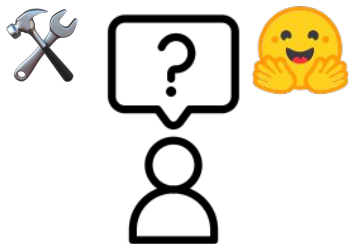
"With your support on  
the installation bug, I  
can focus on making  
scientific discoveries!"



<https://timoelliott.com/blog/cartoons/artificial-intelligence-cartoons>

# StopGap: Building intelligent agent for the installation problems

## Solving research software installation problems with generative models



“I want to convert video file format”



Generative model of prompt-to-code

**Data for Task Automation**

**Instruction**  
I have a text file ("demo.doc") with a video URL and a script ("Script.txt") for a voiceover. I need to download the video, extract the audio, add the voiceover to the video using the provided script, and finally combine the extracted audio with an additional audio file ("new.wav").

**Task Steps:**

- Step 1: Extract the video URL from "demo.doc".
- Step 2: Download the video using the extracted URL.
- Step 3: Extract the audio from the downloaded video.
- Step 4: Add a voiceover to the downloaded video using "Script.txt".
- Step 5: Combine the extracted audio with "new.wav".

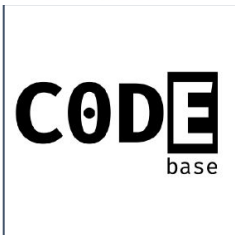
**Tool Invocation Graph:**

### Task-Bench: Task automation

[Yongliang Shen, et al., 2023]  
<https://arxiv.org/pdf/2311.18760>

put a video that people can understand in the readme #375

Open offpress opened this issue 5 days ago · 0 comments



<https://github.com/princeton-nlp/SWE-agent/issues/375>



Generative model of solving Github-issues

```
Problem source: Write here
Between releases 3.0.0rc3 and 3.0.0rc4, Dask Fra Fields has started throwing an error when being instantiated as server fields of container fields like List or Tuple. The
import below works in 3.0.0rc3 and throws the error below in 3.0.0rc4 (link, version: 3.0.0)

python
from marshmallow import Fields, Schema

class MySchema(Schema):
    inner = fields.List(fields.Str())

s = MySchema()
...

Traceback:
...
marshmallow.schema.schema.call next
File "test.py", line 8, in <module>
  s = MySchema()
File "D:\envs\venv\python\marshmallow\lib\python3.8\site-packages\marshmallow\schema.py", line 385, in __init__
  self.fields = self._init_fields()
File "D:\envs\venv\python\marshmallow\lib\python3.8\site-packages\marshmallow\schema.py", line 375, in _init_fields
Local heap path or GitHub URL: https://github.com/swe-agent/dask-fra-fields
```

### SWE Autonomous coding agent

[John Yang et al., 2024]  
<https://swe-agent.com/> @princeton-nlp

# Our goal: verify how well LLMs are able to breakdown the installation problem

Input (readme.md)



Generate



Output (PlanStep.json)

Generative model for  
extracting Plan and Step

**Installation**

**Install from pip**

The key dependencies of DIG: Dive into Graphs are PyTorch (>=1.10.0), PyTorch Geometric (>=2.0.0)

1. Install [PyTorch](#) (>=1.10.0)

```
$ python -c "import torch; print(torch.__version__)"  
>>> 1.10.0
```

2. Install [PyG](#) (>=2.0.0)

```
$ python -c "import torch_geometric; print(torch_geometric.__version__)"  
>>> 2.0.0
```

3. Install DIG: Dive into Graphs.

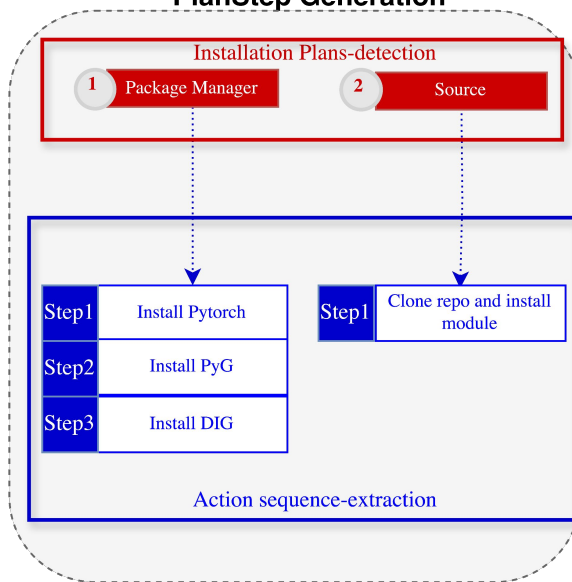
```
pip install dive-into-graphs
```

**Install from source**

If you want to try the latest features that have not been released yet, you can install dig from source.

```
git clone https://github.com/divelab/DIG.git  
cd DIG  
pip install .
```

## PlanStep Generation



```
{  
  "type_plan": "Package Manager",  
  "step1": "Install PyTorch",  
  "step2": "Install PyG",  
  "step3": "Install DIG",  
  "technology": "Pip"  
}  
{  
  "type_plan": "Source",  
  "step1": "Clone repository and install module",  
  "command": "git clone</br>cd</br>pip install ."  
}
```

correct



```
from groq import Groq
```

```
client = Groq()
completion = client.chat.completions.create(model="llama3-70b-8192",
    messages=[
        "role": "user",
        "PROMPT101": "Given the following README, your task is to identify and list the unique installation methods. These methods are plans containing instructions for installing research software, to be executed in a specific order and under defined conditions:\n1. Binary: Install via download and run precompiled files. For example, GitHub releases.\n2. Container: Install the software and its dependencies via isolated environments. For example, Docker, Podman, or Singularity.\n3. Package Manager: Install via tools and indexed repositories. For example, Conda, Homebrew, or Pip.\n4. Source: Run via command-line, manage and install dependencies, compile source code to a target machine, build, and run. For example, download raw source code, clone repositories, and install dependencies from code repositories.\n\nExclude code commands. Be concise.\n\n### {{README:\n\n##}} ###",
        "role": "assistant",
        "response": "Here are the unique installation methods mentioned in the README:\n1. Binary: Not explicitly mentioned, but implied through the mention of PyPI installation.\n2. Container: Not mentioned.\n3. Package Manager: Conda (via Conda Environment Setup)\n4. Source: Installation via GitHub (using `pip install git+https://github.com/KindXiaoming/pykan.git`)\n5. Virtual Environment: Using `python -m venv` to create a virtual environment, and then installing using `pip`.\n\nNote that the installation methods are not mutually exclusive, and some methods may involve a combination of these approaches."
    ],
    temperature=0,
    max_tokens=1024,
    top_p=1,
    stream=False,
    stop=None,
)

print(completion.choices[0].message)
```

P(next word | previous words)

Ziming Lui et al., 2024

Demo from

<https://kindxiaoming.github.io/pykan/>



edu.nl/namwt

from groq import Groq

```
client = Groq()
completion = client.chat.completions.create(
    model="llama3-8b-8192",
    messages=[
        {
            "role": "user",
            "content": "Given the following ### {{README}}, extract the installation instructions for each installation method. These methods are plans containing instructions as steps for installing research software, to be executed in a sequential order, and under defined conditions:\n[DEFINITIONS]\nPerform the following actions. For each README:\n1. Extract the installation instructions for each method .\n2. For each installation method mentioned in the README return a list, where each element of the list is an instruction, in a sequential order. \n3. Write those instructions in sequential order such as: Step1: ..., Step2: .....StepN \n\nExclude code commands in the list. If the README does not contain a sequence of instructions, then simply write \\\"No steps provided\\\". Be concise.\n\n### {{README}} ###"
        },
        {
            "role": "assistant",
            "content": "Here are the installation instructions for each installation method:\n\n**Binary: No installation instructions provided**\n\n**Container: No installation instructions provided**\n\n**Package Manager:**\n\nStep1: Create a virtual environment\nStep2: Activate the virtual environment\nStep3: Install the package via pip\nStep4: Install specific package requirements\n\n**Source:**\n\nStep1: Create a virtual environment\nStep2: Activate the virtual environment\nStep3: Install the package via pip\nStep4: Install specific package requirements\n\nNote: The README does not provide installation instructions for the Container and Binary methods."
        }
    ],
    temperature=0,
    max_tokens=1024,
    top_p=1,
    stream=True,
    stop=None,
)
```

for chunk in completion:

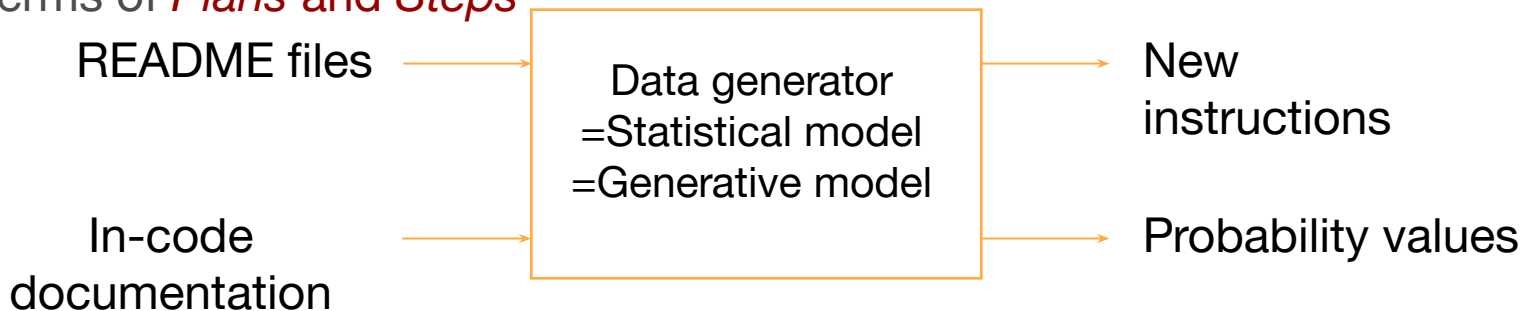
P(next word | previous words)

Ziming Lui et al., 2024

Demo from

<https://kindxiaoming.github.io/p/yan/>

- Develop an approach to verify how well they are able to automatically extract instructions and whether these instructions actually match the ones generated by the original authors.
- LLM to autonomously decide on what sequence of steps to execute to install a software
- Assess the ability of LLMs to capture installation instructions, both in terms of *Plans and Steps*



Automate the extraction and validation of installation instruction using structured framework:

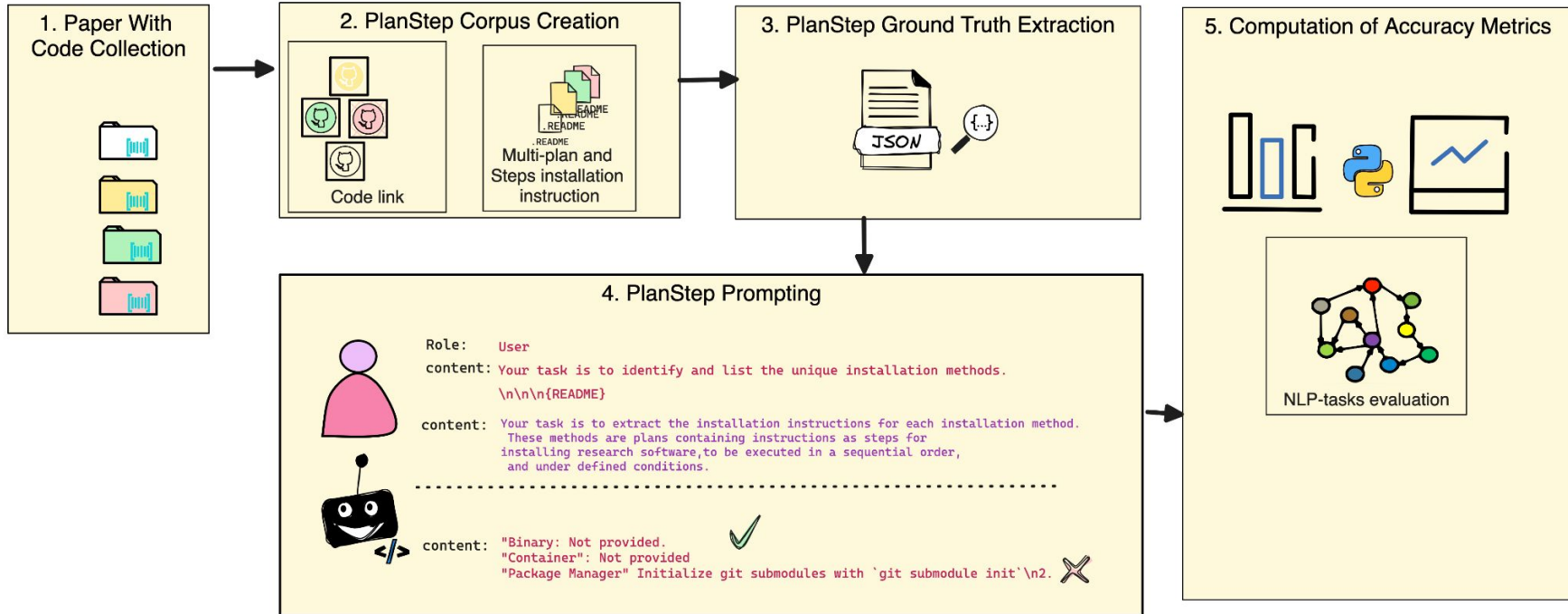


Figure: Overview of the methodology followed to collect research software and design an evaluation framework

33 open-source research software projects from basic installation procedures to more complex



For each README, identify:

## 1. Installation Plans (setup methods)

container



docker compose, run/install from pypi, container, pull image config env

Package Manager



Binary



download and execute files

Source



git clone repo, install dependencies, build the module

## 2. Installation Steps (sequence of actions)

```
{
  "id": "4",
  "name": "utiasASRL/steam_icp",
  "url": "https://raw.githubusercontent.com/utiasASRL/steam_icp/master/README.md",
  "plans": [
    {
      "type": "Container",
      "steps": [
        {
          "text": "Clone this repository and install requirements.",
          "seq_order": 1,
          "is_optional": false
        },
        {
          "text": "Run the container with docker-compose",
          "seq_order": 2,
          "is_optional": false
        }
      ]
    }
  ],
  "README_instructions": "## Installation Clone this repository and its submodules.[...]"
}
```

We defined **Steps** based original's README text





edu.nl/KITags

## We prompt (zero-shot) LLMs (Mixtral and LLaMa) to breakdown the installation problem into two-smaller ones

**1-Plan** prompting: this task is about extracting the installation method as Plans described in a README



**2-Step** prompting: This task asks for detecting detailed, sequential steps for each installation plan found in a README

### Plan Task (PROMPT101)

#### Plan Task (PROMPT101):

Given the following README, your task is to identify and list the unique installation methods. These methods are plans containing instructions for installing research software, to be executed in a specific order and under defined conditions. Exclude code commands. Be concise.

1. Binary: Install via download and run precompiled files. For example, GitHub releases.
2. Container: Install the software and its dependencies via isolated environments. For example, Docker, Podman, or Singularity.
3. Package Manager: Install via tools and indexed repositories. For example, Conda, Homebrew, or Pip.
4. Source: Run via command-line, manage and install dependencies, compile source code to a target machine, build, and run. For example, download raw source code, clone repositories, and install dependencies from code repositories.

Four unique plans and its definition

### Step Task (PROMPT201)

#### Step Task (PROMPT201):

Given the following README, extract the installation instructions for each installation method. These methods are plans containing instructions as steps for installing research software, to be executed in a sequential order, and under defined conditions. Exclude code commands. Be concise.

1. Binary:[...] 4. Source<sup>a</sup>.

<sup>a</sup> We insert the same definitions as states in PROMPT101

**Define the action**  
**Provide context**  
**Some constraints and output specifications**

ASSISTANT

Here are the unique installation methods mentioned in the README:

1. **Binary**: Not explicitly mentioned, but implied through the mention of PyPI installation.
2. **Container**: Not mentioned.
3. **Package Manager**: Conda (via Conda Environment Setup)
4. **Source**: Installation via GitHub (using `pip install git+https://github.com/KindXiaoming/pykan.git`)
5. **Virtual Environment**: Using `python -m venv` to create a virtual environment, and then installing using `pip`.

Note that the installation methods are not mutually exclusive, and some methods may involve a combination of these approaches.

Plan



Step

ASSISTANT

Here are the installation instructions for each installation method:

**Binary**: No installation instructions provided

**Container**: No installation instructions provided

**Package Manager**:

Step1: Create a virtual environment  
Step2: Activate the virtual environment  
Step3: Install the package via pip  
Step4: Install specific package requirements

**Source**:

Step1: Create a virtual environment  
Step2: Activate the virtual environment  
Step3: Install the package via pip  
Step4: Install specific package requirements

Note: The README does not provide installation instructions for the Container and Binary methods.

Example from <https://kindxiaoming.github.io/pykan/>

## Installation

Pykan can be installed via PyPI or directly from GitHub.

### Pre-requisites:

```
Python 3.9.7 or higher  
pip
```

### Installation via github

```
python -m venv pykan-env  
source pykan-env/bin/activate # On Windows use `pykan-env\Scripts\activate`  
pip install git+https://github.com/KindXiaoming/pykan.git
```

### Installation via PyPI:

```
python -m venv pykan-env  
source pykan-env/bin/activate # On Windows use `pykan-env\Scripts\activate`  
pip install pykan
```

### Requirements

```
# python=3.9.7  
matplotlib=3.6.2  
numpy=1.24.4  
scikit_learn=1.1.3  
setuptools=65.5.0  
sympy=1.11.1  
torch=2.2.2  
tqdm=4.66.2
```

After activating the virtual environment, you can install specific package requirements as follows:

```
pip install -r requirements.txt
```

**Optional: Conda Environment Setup** For those who prefer using Conda:

```
conda create --name pykan-env python=3.9.7  
conda activate pykan-env  
pip install git+https://github.com/KindXiaoming/pykan.git # For GitHub installation  
# or  
pip install pykan # For PyPI installation
```



**Plan:** both models achieved around 59% F1-score, but LLAMA significantly outperformed MISTRAL

LLM	Zero-shot		
	Precision	Recall	F1 score
llama-2-7b-chat	0.4615	0.8333	<b>0.5941</b>
Mixtral-8x7b-Instruct-v0.1	0.4068	0.6667	0.5053

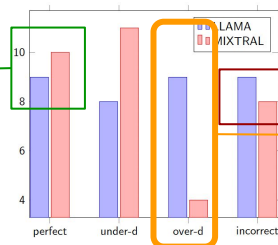
LLM	Zero-shot		
	R1 ↑	R2 ↑	RL
llama-2-7b-chat	29.48	18.88	27.75
Mixtral-8x7b-Instruct-v0.1	46.42	37.53	45.27

**Step:** LLMs demonstrate similarly poor performance (ROUGE) when extracting steps in a plan in order.

Error Type:	LLAMA				MISTRAL			
	E1	E2	E3	O	E1	E2	E3	O
Binary	0	0	1	0	0	0	0	1
Source	15	4	8	1	0	6	4	2
Package Manager	0	0	1	0	1	0	4	0
Container	0	1	1	1	0	0	1	0

Error analysis: lack of context

Number of steps agrees with ground truth



LLMs detected plans and steps falsely

Definitions into the model responses. We need to keep refining with better prompts!

- Interpreting and rigorously verifying installation instructions remain complex for LLMs
- **PlanStep** is a simple yet effective **validation framework** for evaluating LLMs in extracting installation plans and steps
- **Experiments** indicate LLMs struggle with the extraction but show potential
- Future **improvements**:
  - Enhance PlanStep corpus with level of complexity in the installation plans
  - Incorporate novel prompt methods e.g., few-shot-CoT, LLM4PDDL, CoSm
  - Represent instructions in formalised/machine language e.g., P-PLAN, PDDL

Ack:



Corpus



edu.nl/namwt

LLMs output



edu.nl/k7agd

Validation



edu.nl/ej7ed